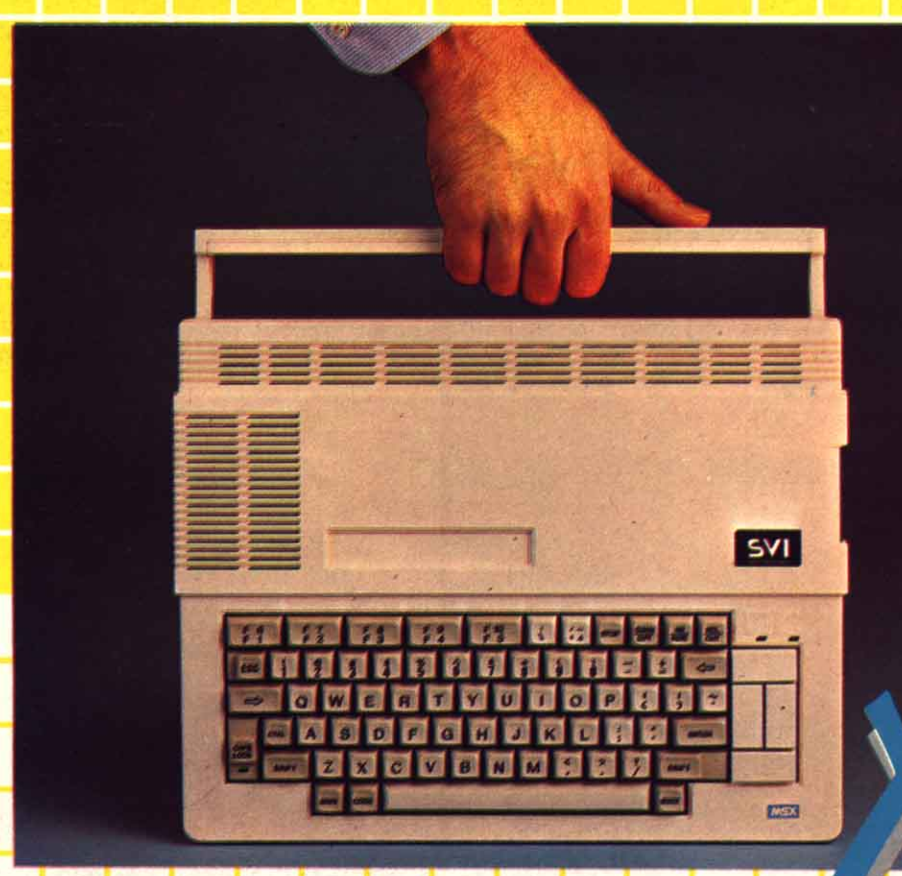

MAGAZINE
MSX
AÑO I
Núm. 7
Noviembre
1985
250 Ptas.
ANALIZAMOS
EL GENERADOR
DE SONIDO
Aplicaciones
matemáticas
y el ordenador
La misteriosa
memoria
de video
Noticias
Sonimag
Programas, etc.

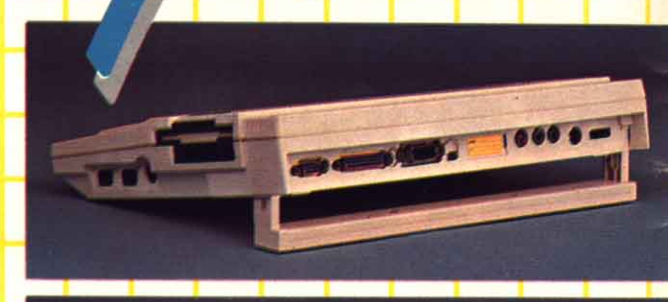
COGE EL X'PRESS





X'press

- Ordenador Sistema MSX
- 80K RAM
- Unidad de disco de 3,5" integrada en el teclado
- Trabaja en CP/M, MSX—DOS, MSX—DISK BASIC
- Teclado profesional de diseño ergonómico. Va incluído un maletín para la protección del ordenador durante su transporte
- Dos puertas de conexión: RS 232—C y Centronics paralelo
- Salida a T.V. y monitor
- Admite directamente una segunda unidad de disco
- MVDP (pasa de 40 a 80 columnas en pantalla. Indispensable para trabajar en CP/M)











Editorial

Este mes tenemos a la vista dos ferias importantes. El SIMO y la Micro Feria del Corte Inglés. En ellas esperamos poder contemplar los adelantos realizados, tanto a nivel de la máquina como de aparición de nuevos periféricos para el estándar.

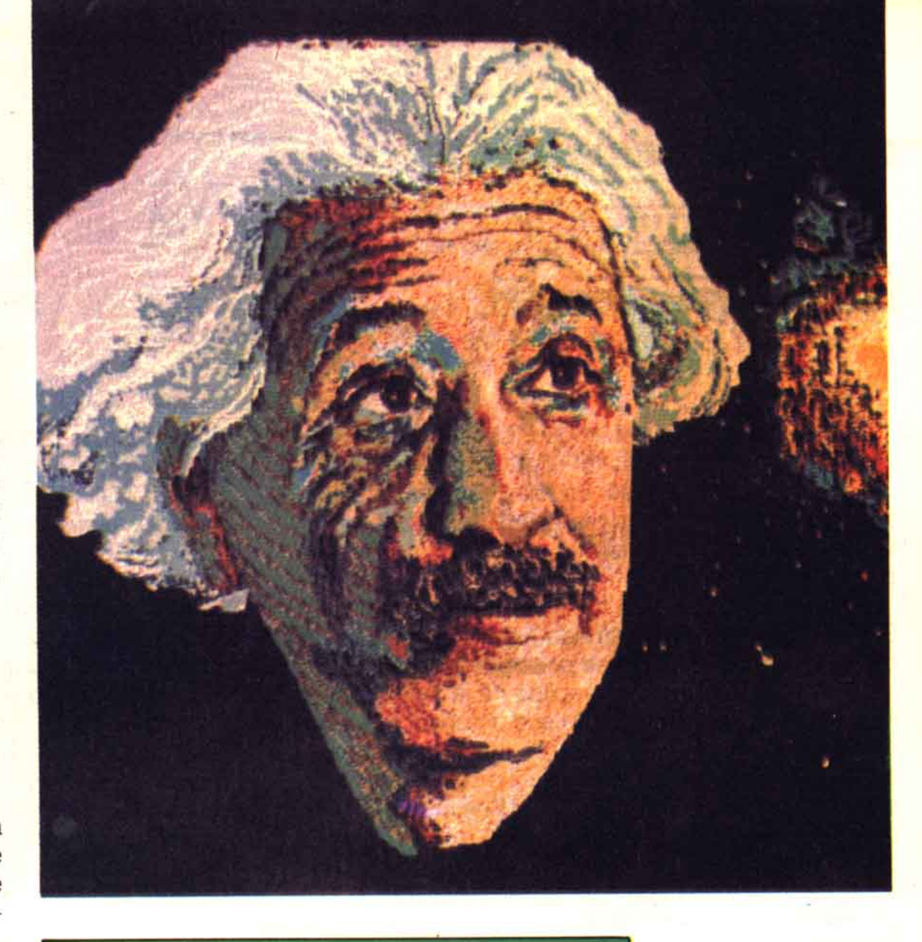
Pero sin lugar a dudas ha sido en la feria de SONIMAG de Barcelona donde hemos tenido ocasión de contemplar las increibles posibilidades del MSX.

En los dos stands más amplios de la feria (Sony y Philips) se expusieron no sólo sus ordenadores más comercializados, sino que ambos nos sorprendieron con una pre-maqueta de lo que será el MSX de la segunda generación: por parte de Sony, el HB-550P y por parte de Phlips, el VG-8240. Pocos detalles y características técnicas pudimos reunir, pero podemos adelantar que estos aparatos de la «segunda generación» se comercializarán a mediados del próximo año, y el precio oscilará entre las 100.00 y las 150.000 ptas. Parece obvio que, por este precio, se nos ofrecerá un ordenador bastante competente, pero todavía es un poco pronto para emitir nuestro juicio.

Otro ordenador que hará su aparición en nuestro mercado es el Pioneer PX-7. La fecha más probable es la de principios del 86. Este ordenador es el primero del estándar que en su versión básica incorpora una unidad de superposición de vídeo a un precio bastante reducido.

El otro aspecto a detacar del SONIMAG ha sido la proliferación de periféricos, prueba de ello es la aparición de un cassette con búsqueda automática de programas, de Sanyo; una unidad de diskettes de 5,25 pulgadas, también de Sanyo; un teclado musical para el Toshiba o una reducida unidad de diskettes de 3,5 pulgadas.

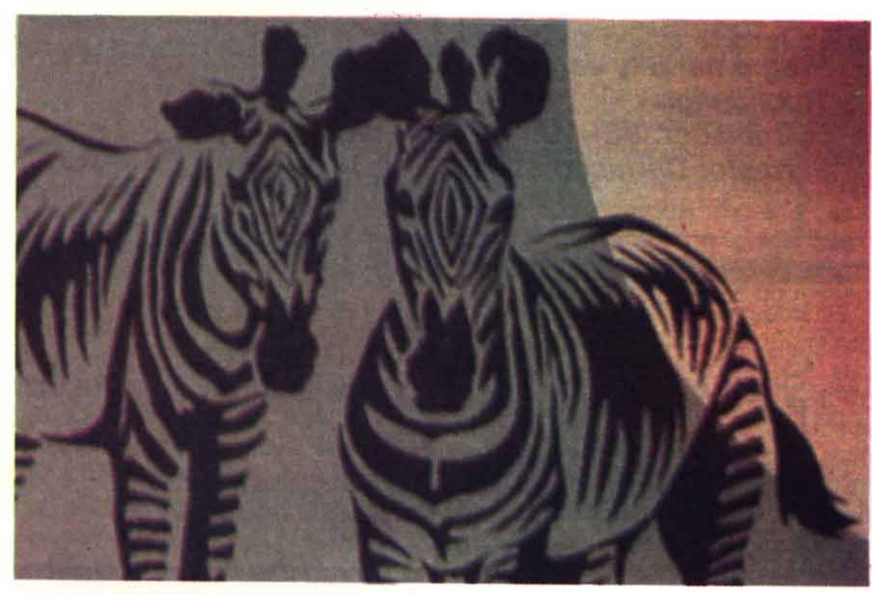
Está claro que los japoneses se han lanzado a la conquista silenciosa del mercado español, para lo que cuenta con una creciente cantidad de software y, en un breve plazo de tiempo, con la publicación de numerosos libros que ayudarán a todos los usuarios de MSX sacarle el máximo rendimiento a su ordenador.



SUMARIO













SOFTWARE

Programa: Sky Jaguar
Tipo: Juego
Distribuidor: SERMA
Formato: Cartucho

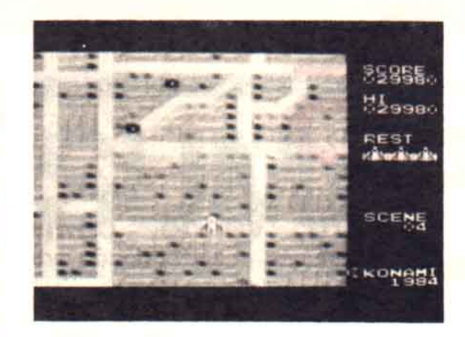
Estamos en el «Cero Punto Cero», primer año civil de la Era de las Galaxias. No, no es que hayamos cambiado el calendario normal, sino que ahora por unos momentos no seremos nosotros, ni viviremos en nuestra era, estaremos en nueva era de color, sonido e imagen de la cual Konami es el creador.

«Jaguares del espacio» no es un simple juego de marcianitos, es una apasionante y trepidante lucha por salvar a la Tierra de los invasores. Es un mundo en que la velocidad del sonido se ha superado, nos moveremos a lo largo y ancho del Universo con la ayuda de nuestro ordenador y ¡claro está!, con nuestra habilidad que nos hará sin duda ser un verdadero *Sky Jaguar*.

Aquellos de nosotros que pasamos la mayor parte de nuestros ratos de ocio, jugando a ser verdaderos defensores galácticos o el más intrepido escalador, veremos colmados estos momentos con este singular juego de Konami. La acción se desarrolla en nuestra galaxia, la Vía Láctea y la Tierra, hoy en día, unificada, busca lugares en los que poder obtener materias primas para poder sobrevivir a los despiadados ataques de los invasores de la nebulosa *Zeifart*, los cuales quieren apoderarse de la Tierra y destruir la Paz que tantos años ha acompañado a los habitantes de ésta.

Nuestra pequeña nave, equipada con los mayores avances de la guerrilla galáctica se verá más de una vez en dificultades, ya que nuestros poderosos enemigos utilizan armas igual de sofisticadas y más variadas. Nos encontraremos a nuestro paso por los satélites del Planeta Urano con lugares devastados, en los que nuestra pericía y valentía de jaguares del espacio tendrán que salvarnos de una total destrucción para devolver a la Tierra su supremacía en el espacio.

Nuestro viaje galáctico se desarrollará por cinco pantalla sucesivas en las de nuestros enemigos, cada vez más poderosos y rápidos, nos intentarán eliminar con sus naves espaciales, con sus bombas, misiles y cargas que al colisionar contra ellas nos harán desaparecer en mil pedazos. Pero dentro de estos tres tipos genéricos, encontraremos a su vez múltiples divisiones de elementos cuyo fin primordial es destruir la Tierra y todo elemento viviente que la defienda.



Es un juego apasionante ya que no encierra ninguna complicación a la hora de ejecutarlo, pero sí implica una gran concentración y astucia digna del más hábil de los pilotos aéreos.

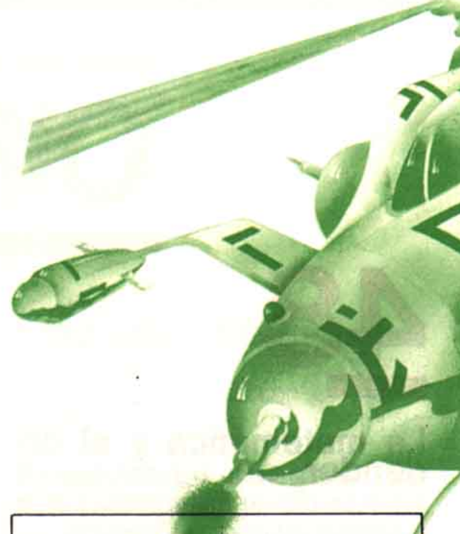
Es un juego diseñado para una sola persona, es decir nuestro adversario será el siempre temible y por supuesto amigo, ordenador. Pero no debemos dejarnos impresionar por todo este mundo vertiginoso que se nos presenta, las naves con las que contamos serán en un principio tres, pero si conseguimos 10.000 puntos contaremos con una nueva nave, y así sucesivamente cada vez que consigamos 40.000 puntos a lo largo de la partida.

No sólo nuestro objetivo serán los invasores, sino todos aquellos complejos que les sirven de ayuda para repostar y lanzar al espacio nuevas naves como elementos aniquiladores. Tendremos por tanto que dirigir nuestro ataque hacia esos objetivos, que se nos presentarán en gran número y sin piedad alguna.

«Los jaguares del espacio», es por tanto una trepidante aventura que nos hará vivir un sin fin de peligros y, lo más importante, un entrenido rato.

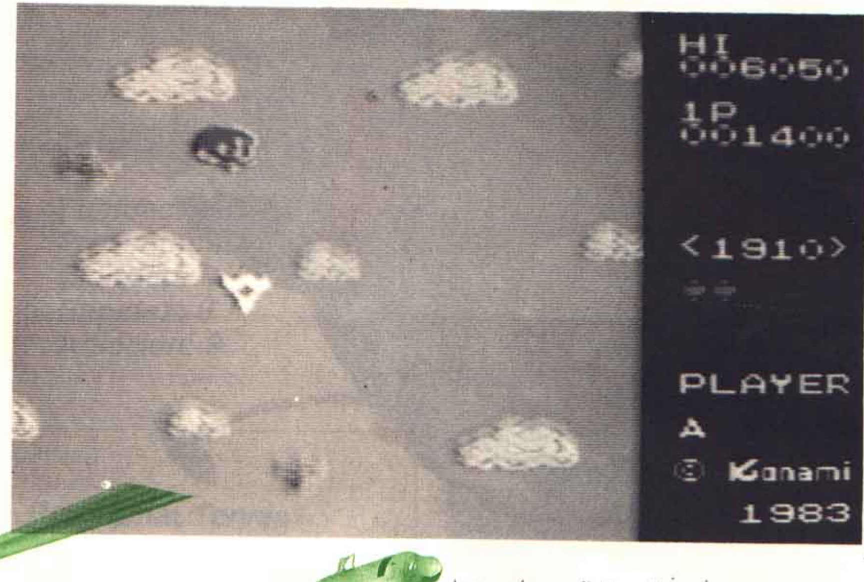
Sus imágenes y color le hacen un clásico entre los clásicos, pero llevando consigo las más depuradas técnicas de los juegos de ordenador.

Puntución:
Presentación: 9
Claridad: 8
Rapidez: 8
Adicción: 9



Programa: Time Pilot
Tipo: Juego
Distribuidor: SERMA
Formato: Cartucho ROM

Las guerras siempre han contado con un elemento primordial que es la aviación, pero disfrutar de este acontecimiento no implica que nos tengamos que encontrar bajo el fuego enemigo, ni siquiera próximos a un



campo de batalla, tan solo es proponerse pasar un rato sentado frente a su ordenador y compartir con él la incertidumbre y el riesgo de verse asediado por los más audaces pilotos, como es el caso de *Time Pilot*.

Piloto del tiempo, es un juego en que los amantes de la aviación, las ametralladoras aéreas y las piruetas en el aire efectuadas por los aviones, pasarán sin duda un rato ameno y muy distraído.

Las distintas Eras de la aviación se verán claramente reflejadas en este juego, ya que a medida que transcurre este, se irán sucediendo los más diversos prototipos utilizados tanto en la 1ª y 2ª Guerra Mundial, como las más sofisticadas armas actuales y aquellas que la imaginación crea para un futuro.

Konami, con este juego, no ha roto su cualidad de innovador, pero se ha mantenido en una línea de claro matiz conservador, es más, ha cuidado el significado, dándole un carácter didáctico al mostrarnos los distintos tipos de aviones que han surcado el espacio aéreo mundial, pero sin dejar de lado su principal labor, que es la de entretener y divertir.

El juego se dividirá en tres etapas principales, una primera en la que los aviones serán los conocidos biplanos de la 1ª G.M. los cuales no nos será nada fácil destruir porque cuentan con una gran destreza y habilidad.

La segunda etapa o Era se referirá a los biplanos, aviones de dos motores que sin duda serán más rápidos que los anteriores. La tercera Era ya se localiza en nuestro tiempo 1984, y aquí el prototipo será el Helicóptero que nos atacará de las más insospechadas formas, puesto que aquí desempeñan un papel importante los misiles.

El paso entre las diversas etapas se hará logrando eliminar un elemento que aparecerá en cada una de ellas en el momento en que hayamos conseguido derribar un número de aviones estipulado por la máquina. En el primer caso será un dirigible al que deberemos de acertar el mayor número de disparos hasta que explote. En la segunda será un avión de carga superior en tamaño a los que constantemente estamos derribando y en la tercera aparecerá un helicóptero de dos hélices de transporte. Una vez conseguido batir el número de aviones estipulado en cada etapa y aniquilado su respectivo dirigible, helicóptero, etc, conseguiremos pasar a la siguiente etapa.

Pero..., existe una cuarta etapa, la cual la dejaremos a usted que la descubra por sí mismo.

Comenzaremos con un número de tres naves, e iremos sumando puntos cada vez que logremos batir un avión o capturar un paracaídas, consiguiendo mayor número de puntos cada vez que pasemos de una era a otra.

Quizá parezca a primera vista un «típico», pero cuando usted juegue con él sin duda que lo pasará de lo más entretenido.

Time Pilot de Konami, es para los amantes de la aviación un entrenimiento que le ocupará más de un instante en sus ratos de ocio y para aquellos que disfrutan con los juegos de ordenador, es el «juego que no puede faltar».

Puntuación:
Presentación: 6
Claridad: 7
Rapidez: 8
Adicción: 7

Programa: Hiper Sport II
Tipo: Juego
Distribuidor: SERMA
Formato: Cartucho ROM

Nos habíamos referido ya de una manera muy concisa a las caracterís-

ticas del *Hiper Sport I* en el anterior número, pero hoy vamos a comentar su segunda parte, una parte dedicada al tiro con arma y arco y al tan duro y sacrificado deporte del levantamiento de pesas o halterofilia.

No es que este tema de juegos sirva para hacer una sucesión de programas, es simplemente que Konami ha querido dar un matiz individual al tema del deporte y no agruparlos todo en un mismo programa.



Por eso en este *Hiper Sport II*, se tratan otros deportes con el mismo carácter de competición y con el escenario adecuado a cada una de estas modalidades, las cuales cuentan con un gran número de aficionados, y lo más importante, de deportistas.

¿Quién no ha jugado a disparar o ha tenido un arco? y es más ¿quién no tiene un juego de pesas en su casa?, todos estos deportes, sin duda, hay que vivirlos. Pues bien, Konami ha encontrado la forma de hacernos vivir la tensión de la superación y el ejercicio desde nuestro ordenador, además visto desde una perpectiva de entretenimiento y diversión.

Hiper Sport II, se subdivide en tres fases eliminativas. La primera dedicada al deporte del tiro de escopeta. Aquí, usted, será un hábil tirador que deberá intentar conseguir el mayor número de platos (pichones de barro), que aparecerán en su pantalla tanto por la derecha como por la izquierda.

Derribar tantos como pueda será su propósito. El ordenador, como ayuda complementaria para facilitarle su labor, dirigirá automáticamente la elevación de su escopeta y sólo tendra que pulsar el botón de disparo cada vez que un plato se encuentre en su ángulo de tiro.

Una vez que hayamos practicado algunas veces seremos unos verdaderos campeones en esta modalidad.

La segunda prueba, también de puntería, se refiere al sofisticado deporte del tiro con arco, pero con un pequeño inconveniente, el blanco o diana sera móvil, y el viento también será elemento a tener en cuenta.

Tanto en la primera como en la segunda modalidad de tiro tendremos tres oportunidades de superar la marca prestablecida por el ordenador; así, en el tiro con arco contaremos con ocho flechas en cada una de las tres oportunidades, con lo que conseguiremos sin duda al menos una diana.

«Halterofilia». No existen palabras que contengan el significado de este gran deporte. El esfuerzo físico, la concentración y la fuerza de voluntad es difícil de medir objetivamente, pero podremos ver de cerca todos estos esfuerzos a la hora de intentar conseguir que nuestro deportista logre subir las pesas, ya que tendremos que pulsar el botón de disparo tantas veces y tan rápido como nos sea posible. Es por tanto que Konami nos ofrece todos los tipos de categorías de pesos, debiendo elegir la más apropiada a nuestra habilidad específica.

Cuando consigamos que los tres jueces enciendan su luz, hemos culminado la prueba.

Este tipo de deporte calificado de doméstico nos ayudará a pasar una tarde entretenida. El ambiente depor-

tivo, la animación y la tecnología de los juegos de Konami, entre la que debemos de destacar la magnífica visualización de los objetos y las personas, hace que *Hiper Sport II*, sea nuestra «Olimpiada Casera».

Puntuación:
Presentación: 9
Claridad: 9
Rapidez: 9
Adicción: 9

Programa: Tennis
Tipo: Juego
Distribuidor: SERMA
Formato: Cartucho ROM

El tenis, juego originario francés, empieza a cobrar espectacularidad alrededor de 1873, cuando las colonias inglesas y la propia Gran Bretaña, comienzan a hacer de él un juego de élite.

Los grandes torneos, como Wimblendon, Rolland Garros, o nuestro gran open Conde de Godó, sin duda no se verán nada desmerecidos tanto en el aspecto profesional como en la espectacularidad que conlleva, al dedicar nuestros ratos de ocio a este gran juego que es el «Tennis de Konami».

Konami sigue estando a la altura de los campeones, ha sabido llevar al ordenador los momentos de tensión, la vitalidad y el derroche de intuición, velocidad y habilidad que caracterizan un encuentro de tenis.

No vamos a describir las reglas del juego, porque de todos son conocidas, pero sí nos vamos a referir a las modalidad en que podemos jugar.

El partido se desarrolla en el escenario propio de una pista de tenis, con sus gradas, jueces y niños «recoge-pelotas», y por supuesto con los ases del tenis entre los que por uno minutos nos encontraremos.

Podemos jugar, como primera opción contra el ordenador, el cual se nos presentará como un duro adversario difícil de batir. Tendremos una segunda opción en la que podremos jugar con un compañero, es decir un individual en que el que dos personas serán los protagonistas del encuentro, y una tercera opción que sin duda a juicio de nuestros especialistas de juegos es la más divertida, el dobles. Aquí jugarán dos personas contra el ordenador, que también serán dos, y donde desarrollará un papel muy importante la rapidez y compenetración entre los jugadores. Es necesario sin duda que antes de comenzar a jugar a esta modalidad establezcamos con nuestro compañero muchas y variadas tácticas de juego, por si las sorpresas.

En la primera y tercera modalidad podemos elegir el ritmo de juego que queremos desarrollar, entre los niveles 1, 2 ó 3, siendo el tercero un nivel que requiere una técnica muy depurada.

Al principio encontraremos alguna dificultad a la hora de golpear la pelota, ya que la perspectividad creada en el campo, nos obligará a colocarnos en ángulo de su bote. También al principio nos será difícil crear un efecto, quizá cuando consigamos esto nos recuerde a un partido de tenis de mesa, pero el escenario en que nos moveremos está referido completamente al deporte del tenis como tal.

Con este programa no sólo vamos a pasar una entretenida tarde jugando al tenis, sino que a medida que se vaya desarrollando la partida nos encantará la presentación que ha hecho de él Konami.

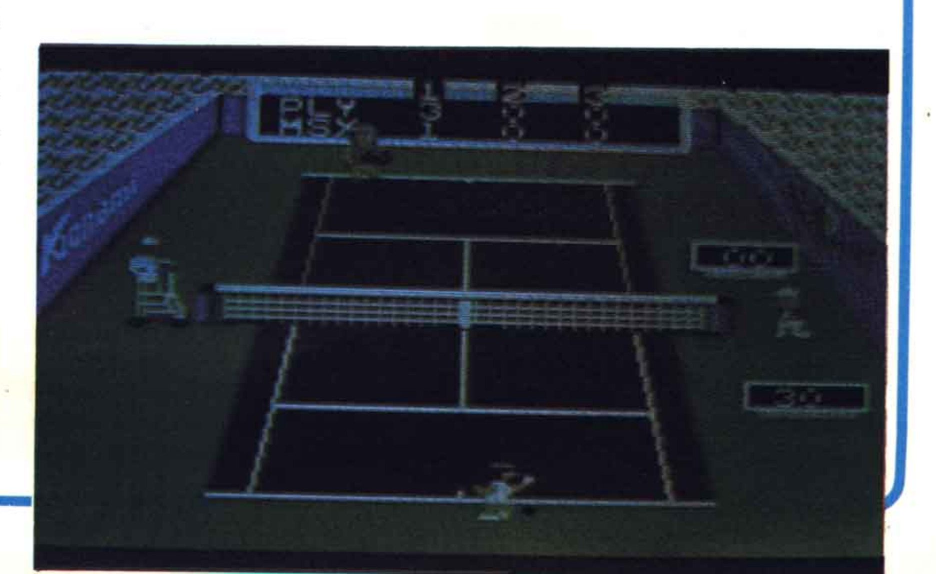
Los jugadores serán lindas señoritas que con su atuendo típico de grandes campeonas nos divertirán con sus movimientos y originalidad.

Contaremos con un juez imparcial el cual arbitrará a lo largo de la partida y un niño «recoge-pelotas» que facilitará nuestra labor.

Los marcadores se encontrarán al final de la pantalla y el «macht» constará de tres «sets»; aquél que consiga dos habrá ganado.

Los colores de la pantalla no serán gratos de visionar ya que la pista al ser de tierra batida será marrón y nuestras deportistas llevan colores claros. El sonido como es característico de Konami está perfectamente delimitado, ya que como este deporte requiere un gran silencio de concentración, no incurrirá constantemente, pero sí a la hora de notar una victoria, o en el peor de los casos una derrota.

Puntuación:
Presentación: 9
Claridad: 8
Rapidez: 8
Adicción: 9



SOFTWARE

Porgrama: Aritmo
Tipo: Didáctico
Distribuidor: DIMension NEW
Formato: Cassette

Este programa a diferencia de otros muchos existentes en el mercado, no se ha contentado tan sólo con dedicarse a una función mera y exclusica como el ocio, el cálculo o la contabilidad, sino que fundamentado en unas bases puramente didácticas, ha sabido entrelazar de una forma ingeniosa la enseñanza y la diversión en todo el contenido didáctico que conllevan cada una de estas dos actividades.

Vamos a dirigirnos a un sector de la población al que específicamente va indicado este programa, que son los «pequeñines», aquellos niños que se encuentran en edades de cinco a ocho años a los que les servirá principalmente a la hora de realizar sus estudios.

¿Cuántas veces nos hemos visto intentando lograr que nuestro hijo permanezca más de cinco minutos intentando efectuar una suma? y ¿cuántas veces nos hemos ingeniado diversiones para estimulrles a ello?

Pues bien este programa ha sido pensado y diseñado como complemento y práctica a las enseñanzas que reciben en clase al motivarles intercalándoles diversiones.

La gran importancia en el ámbito escolar se ve claramente reflejada en este tipo de programas, indispensables para darles a sus hijos una ayuda extraescolar, viéndose altamente favorecida por la aceptación de los niños ante este nuevo método didáctico.

El menú de este programa consta de seis apartados, los cuatro primeros son la esencia del programa, puesto que engloba las operaciones de suma, resta, multiplicación y división; las cuales son en este caso los temas de estudio para nuestro pequeños escolares.

La opción número cinco nos permitirá establecer un recorrido por las tablas del uno al doce, que permitirá a nuestros hijos tener un conocimiento exacto y rápido y una disposición mayor para su comprensión.

Utilizaremos la tecla «A» para avanzar y la tecla «R» para retroceder en una unidad del factor.

Y una sexta opción con un cierto matiz complementario que denominamos «modificar colores», por la que atendiendo a las observaciones del padre o tutor, adecuaremos los colores de la pantalla en conveniencia a quelios que sean más idóneos para una mejor comprensión del alumno. Estos colores irán descendiendo de tonalidad desde el blanco genérico al negro en quince opciones distintas.

En los anteriores apartados referidos a las operaciones de suma, resta, multiplicación y división, el alumno irá cada vez teniendo mayor dificultad en realizar las operaciones, es decir, éstas irán ascendiendo de nivel, basado éste en los principios de la enseñanza Activa, la cual se divide en este caso en nueve niveles. Estos niveles se irán consiguiendo según la capacidad resolutoria del niño, fijándolos la máquina por la capacidad que haya adquirido de resolución.

Si se elige suma, resta, etc., se comprondrá de quince operaciones, finalizadas éstas, el alumno habrá terminado su clase y podrá optar por continuar la clase o pasar unos minutos jugando con los diversos juegos que nos mostrará el programa.

Vamos a comentar cada uno de ellos, entre los que encontramos:

I. ASTEROIDES. Pulsando la tecla correspondiente a este juego veremos a nuestro hijo inmerso en el espacio, dirigiendo o comandando una nave intergaláctica, la cual se moverá por la pantalla del ordenador pulsando los botones del cursor derecha e izquierda.

II. MASTERMIND. Este es un juego creado para que su hijo desarrolle su nivel racional y logístico, al intentar descubrir un número de cinco cifras que el ordenador nos prefijará y el cual podremos hallar con las pistas que el ordenador nos da. Siempre que encontremos la letra «h» a la derecha del número que hayamos pensado nos idicará que ese es el número, pero que está mal colocado. La letra «m» nos indica que hemos acertado número y posición a la vez.

III. ADIVINA EL NUMERO. Aquí será el ordenador también el que piense un número, pero el sistema para acertarlo es diferente. Escribiremos un número entre una y cinco cifras y el ordenador nos dirá si hemos calculado por alto o por bajo hasta que encontremos el número que él ha designado, intentado así coseguir este número en el menor número de probabilidades posibles.

IV. PIANO. Es la única opción que no tiene tope de tiempo, puesto que su hijo podrá crear un número de melodias indefinido.

Esta opción se ha dejado en cierto modo al arbitrio de los padres o tutores para que en determinados momentos puedan alargar el merecido descanso de sus hijos tras la difícil tarea.

Es un programa creado para la ayuda de la enseñanza que tiene como fin el ayudar y divertir a los estudiantes de E.G.B.

Puntuación:
Presentación: 8
Claridad: 8
Rapidez: 8

VENDALE UN PROGRAMA A SONY POR 500.000 PTAS.

Participa en su Concurso de programas para ordenador Hit Bit-MSX.
Sony convoca un Concurso de programas para ordenadores MSX bajo dos categorías.

A – Programas de Contenido Didáctico. Tema de contenido didáctico desarrollado por Centros Docentes entre los especificados en los planes de estudios vigentes.

B – Programas libres. Tema libre desarrollado por usuarios de ordenadores MSX.

Premios

A – Para el mejor programa didáctico.
500.000 ptas. para el Centro Docente
500.000 ptas. para los autores

B – Para el mejor programa de tema libre.
1er. premio 500.000 ptas. para el autor o autores.
5 premios: 100.000 ptas. para cada uno de los siguientes 5 clasificados.

Requisitos

- Los programas presentados por los Centros Docentes deberán tener un máximo de 28 K. RAM.
- Los programas presentados por usuarios deberán tener un máximo de 12 K. RAM.
- Sony tendrá la propiedad de los programas premiados.
- Sony tendrá los derechos de compra sobre el resto de los programas presentados.
- Los programas que concursen deberán ser presentados grabados en cinta de audio Sony o diskette Sony OM-D3440, entregándose dos copias. Asimismo se deberá adjuntar un listado, instrucciones de funcionamiento y una síntesis del contenido del programa.
- Con cada programa se entregará un sobre cerrado conteniendo los datos del autor o autores, y en exterior figurará el título correspondiente.
- Todos los concursantes, independientemente de su clasificación final, serán obsequiados con un producto Sony.

Fecha de entrega de los programas

La fecha tope para la recepción de los programas es el 30 de Noviembre de 1.985. Debiendo ser entregados a Sony España, S.A. Departamento Ordenadores MSX. Sabino de Arana, 42-44 08028 BARCELONA. T.– (93) 330 65 51.

Fallo del concurso y entrega de premios

Entre todos los programas recibidos, Sony eligirá los que a su juicio, contengan un mayor nivel de innovación y creatividad.

El fallo se hará público el 29 de Diciembre de 1.985 y publicado en la prensa nacional.

Para mayor información o consulta, diríjase a cualquiera de las Delegaciones Sony.



DELEGACIONES SONY ESPAÑA, S.A.

BARCELONA
Sabino de Arana, 42-44
Tel. (93) 330 65 51
08028 BARCELONA

MADRID
Julian Romea, 8
Tel. (91) 253 08 00
28003 MADRID

BILBAO
Pintor Lecuona, 1
Tel. (94) 444 42 00
48012 BILBAO

SEVILLA
Niebla, 8
Tel. (954) 27 47 07
41011 SEVILLA

VALENCIA
Salvador Ferrandis Luna, 6
Tel. (96) 325 35 06
46018 VALENCIA

LA CORUÑA
Avda. Ejército, 23
Tel. (981) 29 98 55
15006 LA CORUÑA

NOTICIAS

SONIMAG 1985: LA PUERTA DEL FUTURO

Empezar hablando del Sonimag es hacer hincapíe en las aplicaciones, que a diario vivimos, con respecto al tratamiento de imagen y sonido.

¿Dónde mejor que en esta conocida feria para que los fabricantes mostrasen las habilidades de un ordenador conectado a un vídeo?. Indudablemente, esta parece ser la utilización en un futuro no muy lejano del ordenador personal.

determinar pero cuenta con la facilidad de poder conectar un Video Disco a la unidad de superposición de vídeo (que viene incoporada con un lenguaje adicional, el P-BASIC, para el tratamiento del vídeo) y realizar cualquier tipo de composición sobre el vídeo. Este ordenador permite trabajar con el BASIC MSX y el P-BASIC a la vez o solamente con la primera opción.



Un Compact Disc y un MSX van a terminar unidos por un fin común: el tratamiento de la imagen.

El Pioneer PX-7, pionero en esta especialidad, aparecerá en enero del 86 con unas posibilidades nada despreciables aunque tenga 32K de memoria RAM. Su precio está aún sin

Sin embargo, en futuras aplicaciones veremos como el vídeo-disco se convierte en una unidad de almacenamiento masivo, CP-ROM, con capacidades realmente asombrosas donde ésta ya no se medirá en Kbytes sino en Megabytes.

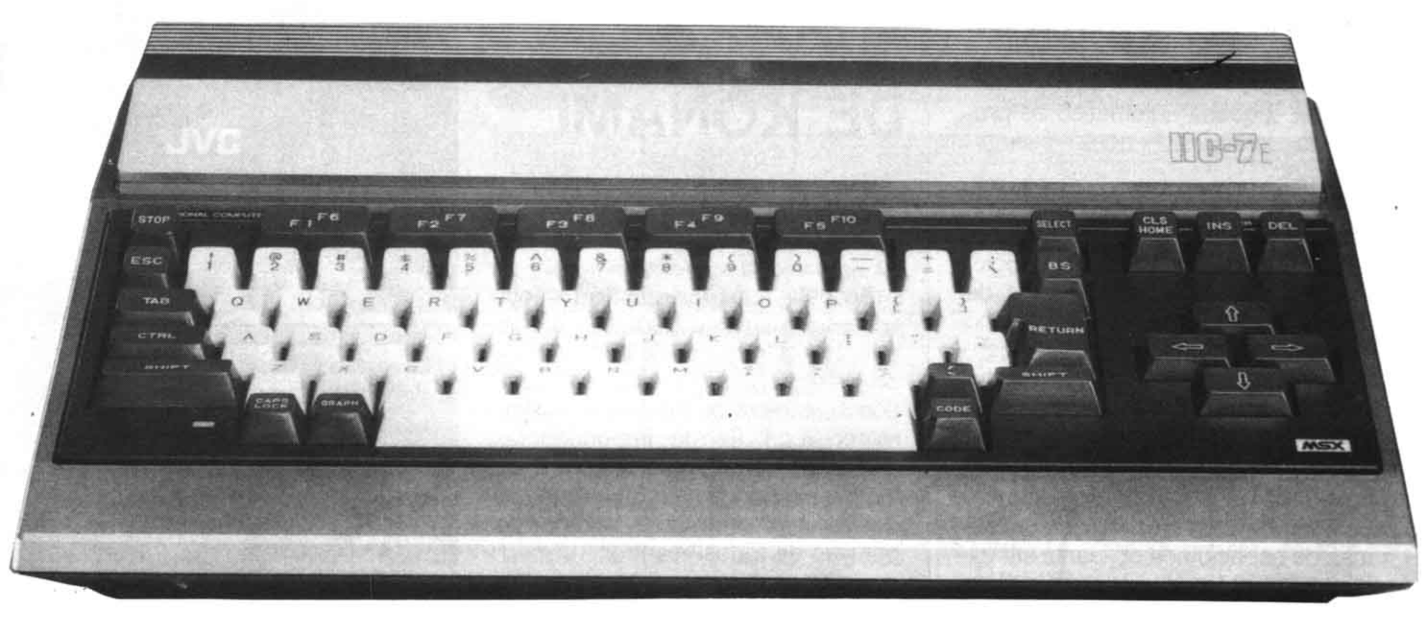
Otra novedad notable corrió a cargo

de dos casas sumamente conocidas por todos los usuarios, Sony y Philips. El primero con sus ordenadores HP-75P y HB-101P, ha acaparado la aceptación general del pequeño público, los chavales se pudieron dedicar a las luchas contínuas con el invasor enemigo. Pero lo más destacado fue la aparición de una premaqueta de lo que va a ser la segunda generación. El HB-500P será la bandera de Sony en lo que a ordenadores MSX de la segunda generación se refiere. Probablemente se comercializará a mediados del año que viene a un precio no especificado por el momento. Este equipo viene con una unidad de superposición de vídeo, 8 modos de pantalla (además de los 4 que vienen originalmente, existirán otros cuatro modos más, con un total de hasta 256 colores) y diversas mejoras en todos los sentidos.



Por su parte, Philips, con sus VG-8010 y VG-8020, hizo las delicias de grandes y pequeños. Sus ordenadores, así como los periféricos que se expusieron (joysticks, la impresora VW-0030 y la unidad vertical de diskettes VY-0010) merecieron una atención especial, puesto que mejoran las prestaciones del estándar. También Philips mostró una premaqueta de su ordenador para la segunda generación. El VG-8240 que incorpora todas las ventajas de los ordenadores de esta generación: 120K VARM, 80 columnas, Diskette incorporado, etc... Se comercializará a mediados del año que viene y su precio oscilará entre 100.000 y 150.000 ptas.

Spectravideo también mostró su nueva versión de MSX, con un teclado cómodo y profesional, diskette incorporado, 80 columnas, port RS-232, y una serie de ventajas que le convertirán en digno competidor de



ordenadores con más renombre.

Toshiba comercializará su HX-20 a partir de Enero. Este reunirá las características típicas de los ordenadores de la segunda geneneración, aunque en la feria nos demostró que no sólo de programas se alimenta el ausuario. Un teclado musical, el HX-MU90, convierte el ordenador en el más completo sintetizador de sonido, con innumerables posibilidades de aprendizaje y aplicación.

Mitsubishi y su ML-GL10 tiene su hueco en el mercado, pero además de actualizar su máquina han presentado una serie de periféricos, tales como un ratón y una unidad de diskettes de doble cara, doble densidad (ML-F30) que es de suma utilidad para aquél que desée darle más utilidad a su ordenador.



Otro gigante, Sanyo, expuso no sólo su ordenador, el PHC-28S, sino que mostró la unidad de diskettes doble de 5.25 pulgadas y un cassette con búsqueda automática de programas como novedades más importantes.

No podían faltar las casas de software y distribuidores, tales como ACE, DIMension NEW, Idealogic, SERMA,... ni los concidos libros de DATA BECKER, con títulos como Metodología de la Programación; MSX, el manual escolar y varios títulos más que comentaremos más adelante en estas páginas.

Estas sólo han sido algunas de las novedades que hemos podido ver y comentar, ya que los lógicos problemas de espacio nos obligan a ser explícitos en el tema.

I SIMPOSIO INTERNACIONAL SOBRE EL CONOCIMIENTO Y SU INGENIERIA

Durante este mes, en la Facultad de Informática de la Universdad Politécnica de Madrid, tendrá lugar el I Simposio Internacional sobre el conocimiento y la Ingeniería, con la participación de venticinco Centros de Investigación de todo el mundo.

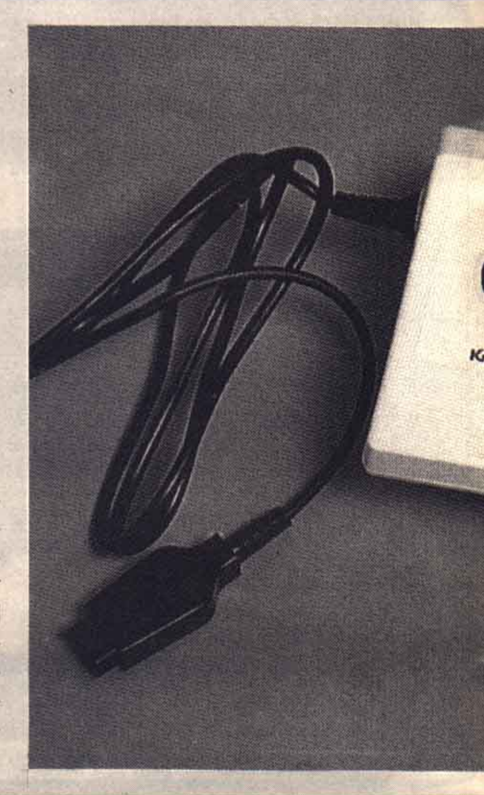
Se darán cita los Centros pioneros de la Inteligencia Artificial, que tratarán temas tan diversos como: el Marketing del Conocimiento, Inteligencia Artificial «versus» Inteligencia Humana, Ingeniería del Conocimiento, Sistemas basados en el Conocimiento, etc.

La Presidencia del Comité de Honor la ostenta S.A.R. El Príncipe de Asturias, el comité está integrado por los ministros españoles de Industria y Energía, Educación y Ciencia y Cultura; el Presidente del Consejo Superior de Investigaciones Cientifícas, el de la Real Academia de Ciencias, los ectores de siete Universidades, el Presidente de la Compañí Telefónica y otras personalidades de ambientes científicos y culturales.

En esta cumbre de Inteligencia Artificial se darán cita más de setecientos especialistas de los cinco continentes.

JOYSTICK ESPECIAL PARA JUEGOS DE KONAMI

Se está comercializando un joystick especialmente dedicado a combatir el mal trato que se le da al teclado del ordenador, en caso de jugar con cualquiera de los programas del Hyper Sport. Serma, importador exclusivo de los juegos de Konami, distribuye este original joystick que facilitará en gran medida las hazañas deportivas de todos vosotros.



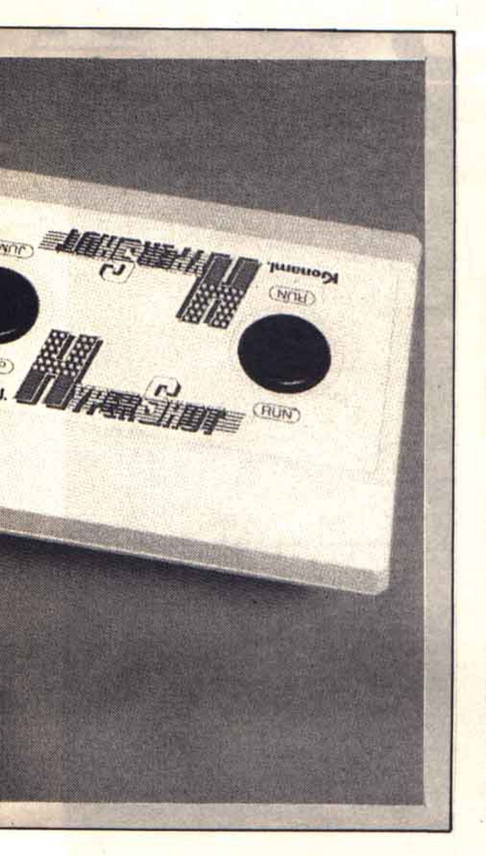
NUEVO MSX DE SPECTRAVIDEO

INDESCOMP acaba de presentar el SV-738 XPRESS. El ordenador tiene una serie de características que le hacen destacar sobre el resto de la gama.

Por un precio de 99.000 ptas. el ususario dispondrá, además de todos los elementos que incorpora el estándar, de una unidad de discos de 3,5 pulgadas integrada al sistema, posibilidad de trabajar en 40 u 80 columnas, interface RS-232C y Centronics y los siguientes sistemas operativos: CP/M, MSX-DOS y MSX DISK BASIC.

También incorpora una asa que le hace transportable y fácil de llevar de un lado a otro, y además una bolsa-maletín de un diseño atractivo, preparado especialmente para protegerlo de los golpes.





NUEVA IMPRESORA PARA MSX

Dirac, conocida empresa dedicada a la fabricación de impresoras, ya posée una especial para MSX. La nueva SP-1000 MSX, es una impresora matricial con una serie importante de ventajas, entre las que podemos destacar la gran variedad de tipos de caracteres, los 86 caracteres RAM programables por el usuario, fijación de márgenes a la izquierda y derecha y opción de copiado de pantalla. Su precio es de 69.900 ptas.

ACLARACION

El mes pasado BUSY hizo de las suyas y alteró el precio de un ordenador.
En el articulo "Los 8 magníficos", el Toshiba aparece con un precio inferior al que realmente tiene, siendo este de 59.900 pts. y no de 53.990 pts. como se indicaba en el artículo.



Anális gener

Cuando los fanáticos de conocer las cosas por dentro abrimos nuestro MSX, aparte de no sacar nada en claro, vemos tres circuitos integrados que destacan por su tamaño: el Z-80, el TMS 9918A y el AY-3-8910. Estos son, respectivamente, el cerebro, el dibujante y el músico de un MSX. Al referirnos a la descripción del estándar MSX, descrito en el manual de MICROSOFT, descubrimos que se hallan bajo la denominación LSI's. Esto es un referencia a la técnica de fabricación empleado en ellos, y quiere decir, básicamente, que son muy complejos.

is del ador de sonidos

En este artículo analizaremos a fondo el AY-3-8910, desde tres puntos de vista: el de sus interioridades, el de sus posibles aplicaciones en el campo de los ordenadores y el de su aplicación concreta en el caso de los MSX.

La arquitectura del PSG

La denominación de PSG nace de las siglas inglesas de *Programable Sound Generator* (Generador Programable de Sonido) que fue el nombre que dio su primer fabricante, *General Instruments,* al AY-3-8910 y a su hermano pequeño el AY-3-8912.

El fin inicial al que iba destinado este circuito era al de generación de efectos especiales para máquinas de bares de producción de sonidos varios en órganos electrónicos.

Posteriormente ha demostrado también una gran utilidad para sonorizar diversos ordenadores y, actualmente, es raro el ordenador personal para el que no se haya editado alguna vez, un artículo llamado algo asi como «HAGA MUSICA CON SU XXX. Montaje utilizando el GI AY-3-89-10».

Es evidente la importancia que tiene y ha tenido este dispositivo en el mundo informático, y todo ello viene de la flexibilidad de su diseño que hace difícil que se quede obsoleto.

La estructura de este circuito puede verse en la figura 1, ya editada en un número previo de esta revista. En ella aparecen tanto los bloques de control y de generación de sonidos como los bloques de conexión con el exterior y las líneas de control y datos que se usa para su comunicación.

En la figura 2 vemos como es el PSG de patas para afuera. A continuación vamos a describir la función de cada una de dichas patas.

DA0-DA7: Forman el Bus de datos para comunicarse con el microprocesador. Sirven para seleccionar el registro de control a modificar y para enviar o recibir su valor, según el protocolo que veremos más abajo.

A8,A9: Sirven exclusivamente para añadir dos bits más de identificación del PSG en sistemas que dispongan de más periféricos. Para seleccionar el PSG deberemos poner A8 en nivel lógico «1» (+5 voltios) y A9 en nivel lógico «0» (0 voltios). Cualquier otra combinación mantendrá inactivo al PSG en su uso de las líneas DA0-DA7.

BDIR,BC2,BC1: Estas líneas indican al PSG el significado del valor puesto en DA0-DA7 o que se quiere leer de él, según la tabla 1. De este modo, a cambio de la complicación mayor del interior del *chip*, conseguimos una gran simplificación tanto de la circuitería exterior como del soporte lógico *(software)* de funcionamiento, lo que redunda en un abaratamiento del equipo.

1OA0-IOA7,IOB0-IOB7: Son las líneas externas de los puertos de en-

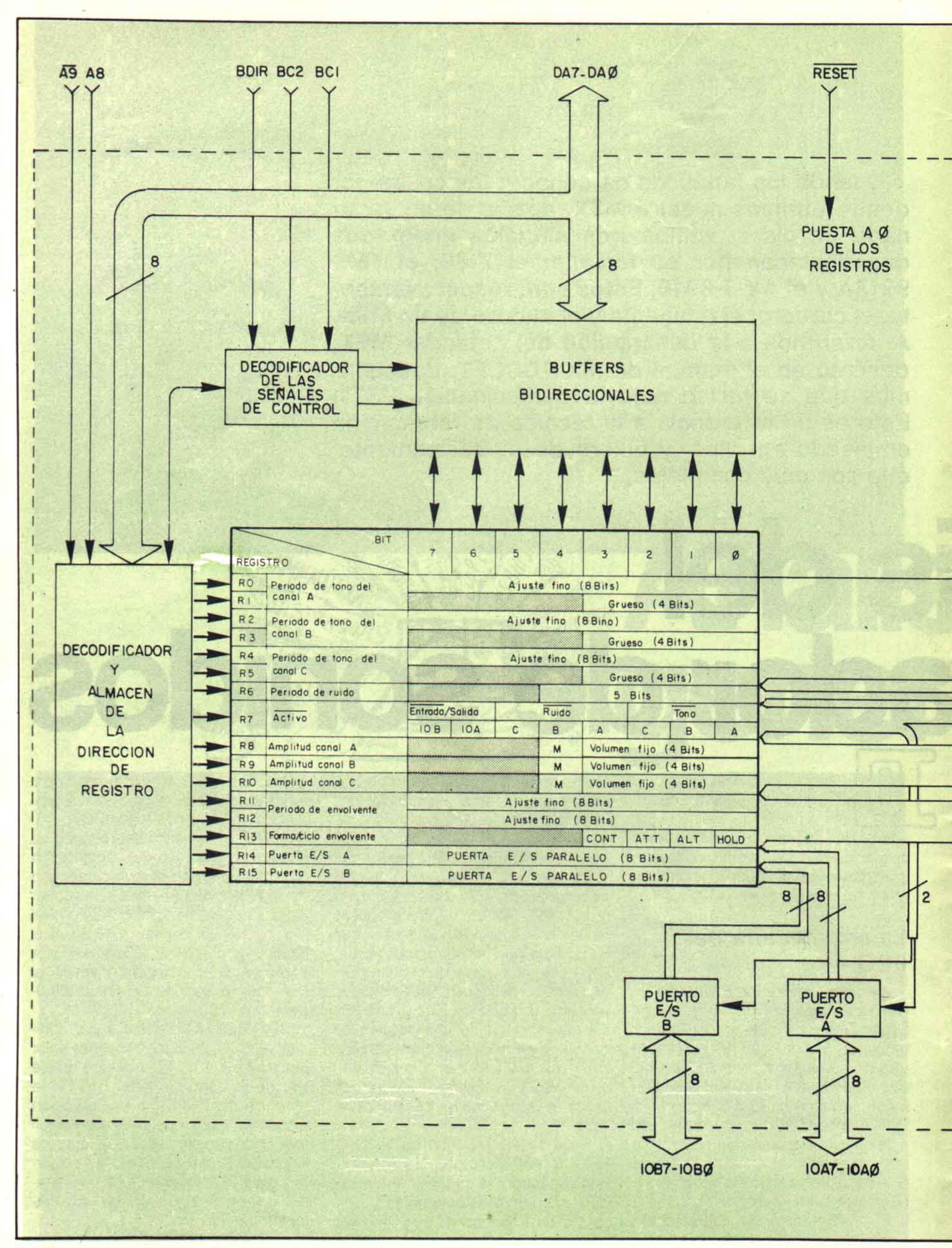
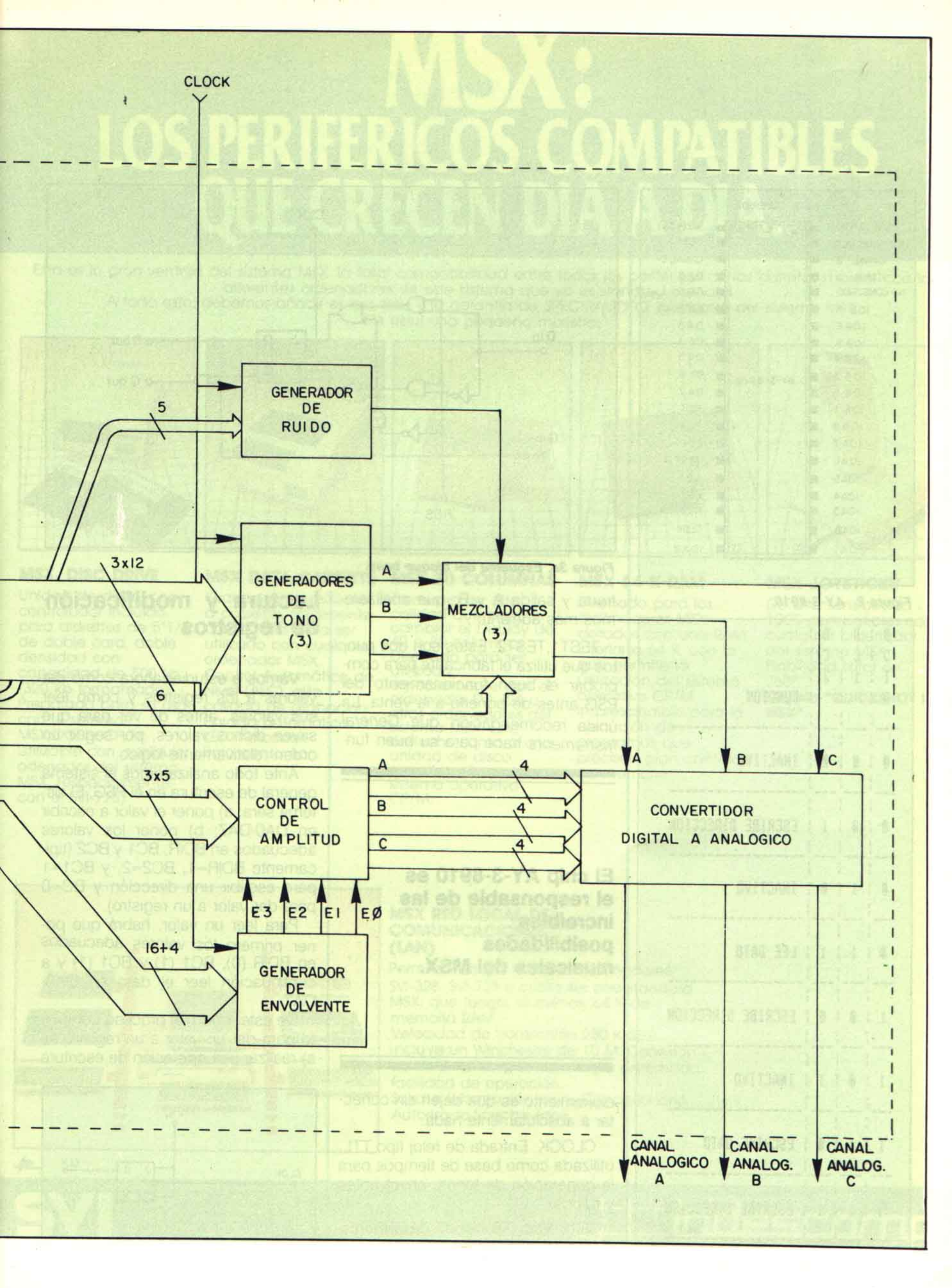


Figura 1. Arquitectura del PSG-3-8910.



CLOCK
GENERADOR DE RUIDO
5
3x12
GENERADORES DE TONO (3)
MEZCLADORES (3)
A
B
C
6
3x5
CONTROL DE AMPLITUD
4
4
4
CONVERTIDOR DIGITAL A ANALOGICO
E3
E2
E1
EØ
16+4
GENERADOR DE ENVOLVENTE
CANAL ANALOGICO A
CANAL ANALOG. B
CANAL ANALOG. C

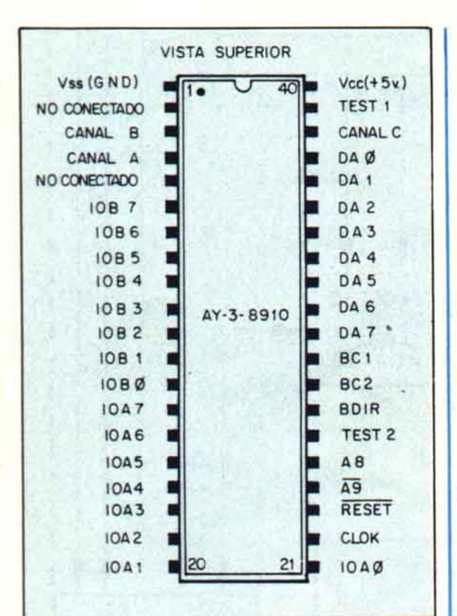


Figura 2. AY-3-8910.



BDIR	BC1	BC2	FUNCION
0	0	0	INACTIVO
0	0	1	ESCRIBE DIRECCION
0	1	0	INACTIVO
0	1	1	LEE DATO
1	0	0	ESCRIBE DIRECCION
1	0	1	INACTIVO
1	1	0	ESCRIBE DATO
1	1	1	ESCRIBE DIRECCION

Tabla 1. Funciones de las señales

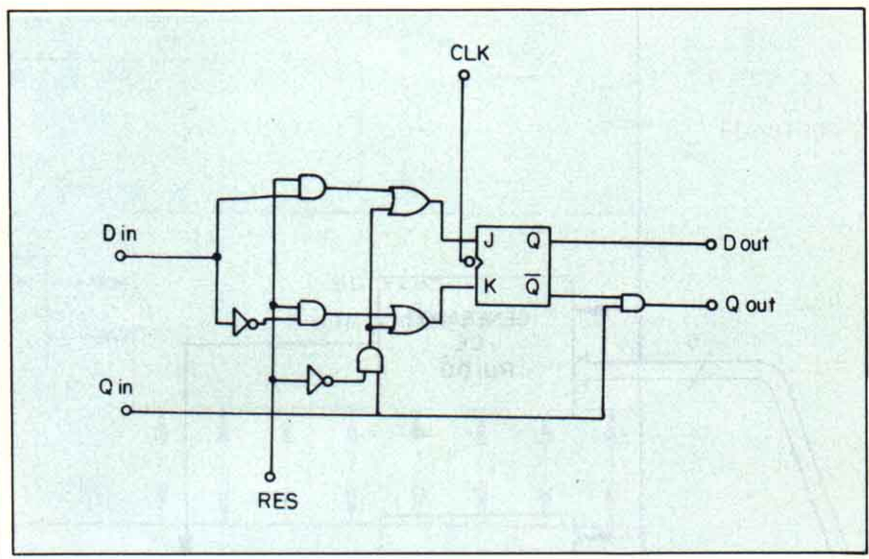


Figura 3a. Esquema del bloque base.

trada y salida A y B, que analizaremos más adelante.

TEST1,TEST2: Estos son dos puntos que utiliza el fabricante para comprobar el buen funcionamiento del PSG antes de ponerlo a la venta. La única recomendación que *General Instruments* hace para su buen funcionamiento es que dejen sin conectar a absolutamente nada.

El chip AY-3-8910 es el responsable de las increíbles posibilidades musicales del MSX.

CLOCK: Entrada de reloj tipo TTL, utilizada como base de tiempos para la generación de tonos, envolventes y ruidos.

Vcc, Vss: Tensiones de alimentación, + 5 y 0 voltios respectivamente.

Lectura y modificación de registros

Vamos a estudiar ahora como dar valores a los registros y como leer sus valores, antes de ver para que sirven dichos valores, por seguir un orden relativamente lógico.

Ante todo analizaremos el sistema general de escritura en el PSG. El sistema sera: a) poner el valor a escribir en DA0-DA7; b) poner los valores adecuados en BDIR, BC1 y BC2 (típicamente BDIR=1, BC2=2, y BC1=1 para escribir una dirección y BC=0 para dar valor a un registro).

Para leer un valor, habrá que poner primero los valores adecuados en BDIR (0), BC1 (1) y BC1 (1) y a continuación leer el dato en DA0-DA7.

De este modo, el proceso completo para dar un valor a un registro es a) realizar una operación de escritura

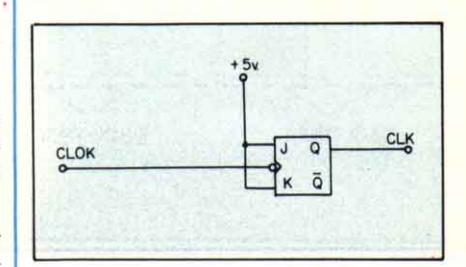


Figura 3b. Esquema de divisor

MSX: LOS PERIFERICOS COMPATIBLES QUE CRECEN DIA A DIA.

Esta es la gran ventaja del sistema MSX: la total compatibilidad entre todos los periféricos de las distintas marcas con los diferentes ordenadores de este sistema que ya es standard mundial.
A todo esto, debemos añadir el respaldo y la garantía de SPECTRAVIDEO, creadores del sistema MSX.
He aquí una pequeña muestra:

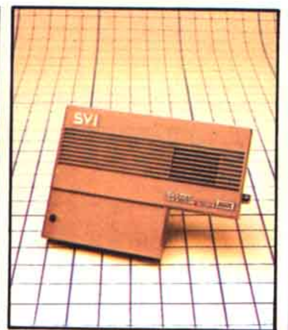


MSX. DISC DRIVE

Unidad de disco, con controlador incluido, para diskettes de 5"1/4 de doble cara, doble densidad con capacidad de 500 kb (320 kb formateado). Preparado para su uso con el sistema MSX-DOS y CP/M. Utilizable con cualquier odenador del sistema MSX. (especialmente con el SVI-728).

MSX DATA CASSETTE

Grabador, reproductor a cassette, totalmente compatible para ser utilizado con cualquier ordenador MSX.
Control Automático de Nivel (ALC), alta calidad de grabación.
Parada automática.
LED indicador.
Bajo consumo.



MSX 80 COLUMNAS

Cartucho de alta calidad que permite cambiar el Display de 40 a 80 columnas. Diseñado para ser utilizado especialmente con el ordenador SVI-728 u otros ordenadores del sistema MSX. Este cartucho, junto a una unidad de disco permite utilizar el sistema operativo CP/M.



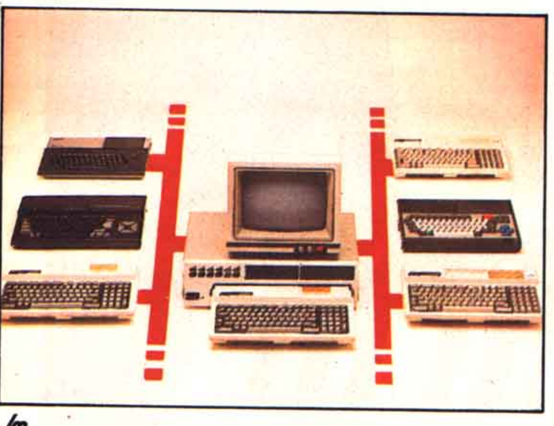
MSX 64 K RAM

Diseñado para los ordenadores MSX dotados con una RAM inferior a 64 K, con lo que permite la utilización del sistema operativo CP/M. Imprescindible para la ejecución de programas que precisen gran cantidad de memoria.



MSX JOYSTICKS

De alta sensibilidad 100% compatibles con cualquier ordenador del sistema MSX. Fiabilidad total en 360°.
Modelo "QUICKSHOT I MSX"



MSX RED LOCAL DE COMUNICACIONES (LAN)

Permite trabajar hasta 32 ordenadores SVI-328, SVI-728 o cualquier otro standard MSX, que tenga, al menos, 64 K de memoria RAM.
Velocidad de transmisión 230 K/SEG.
Incluye un Winchester de 10 M. Conexión del sistema de gran sencillez y extremada facilidad de operación.
Solicite información en su Concesionario Autorizado Spectravideo.







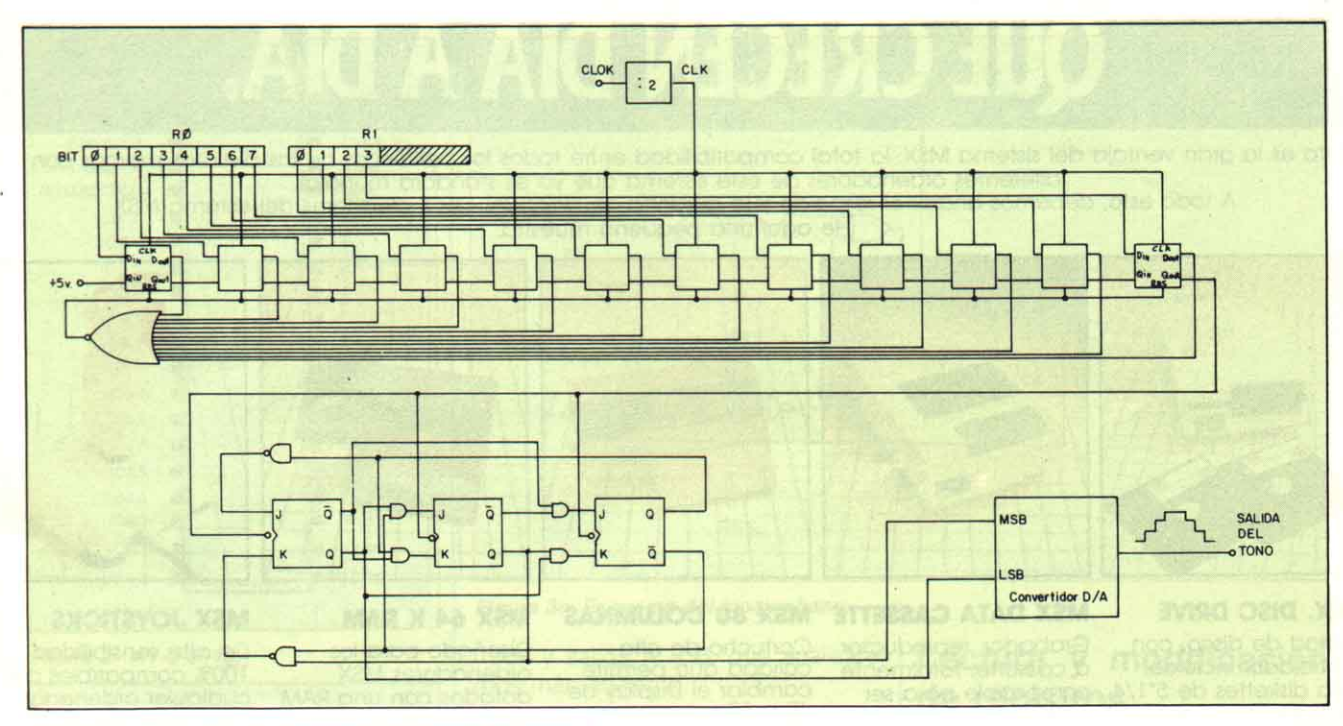


Figura 3c. Esquema del generador de tonos del canal A.

con BC1=1 y DA0-DA3 que sean el número binario que represente el número de registro a modificar. A continuación, realizaremos otra operación de escritura con BC1=0, y DA0-DA7 con el valor a meter en el registro.

Para leer el valor de un registro, se deberá hacer una operación de escritura como la primera que se realizó para escribir en un registro, y a continución una operación de lectu-

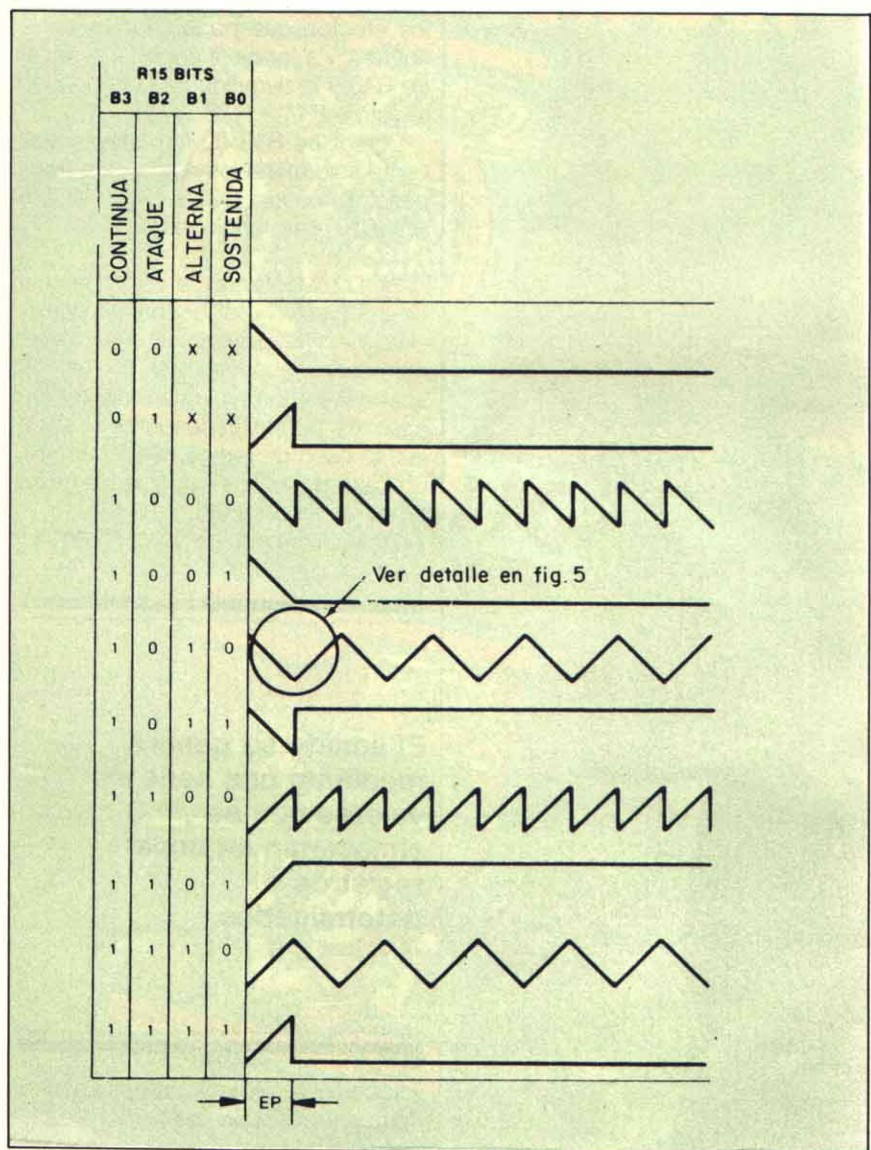


Figura 4. Formas de envolvente en relación con el valor de R15.
— «x» significa que no importa el valor que se dé.
— EP es la duración de la subida fijada por R11 y R12.

ra, leyedo en DA-DA7 el valor del registro pedido.

Generación de tonos

La generación de tonos se realiza mediante dos contadores. El primero de ellos toma como valor inicial el número de 12 bits contenido en el registro de control de tono del canal correspondiente (R0 y R1 para el A, R2 y R3 para el B, R4 y R5 para el C) y va quitándole 1 para cada dos ciclos del reloj externo *(CLOCK)*. Cada vez que este contador pasa por 0, vuelve a tomar el valor del registro de control y genera un pulso.

Dicho pulso se utiliza como entrada para el siguiente contador, que es de dos bits y por tanto puede tomar 4 valores distintos (00, 01, 10 y 11). Este contador suma 1 cada vez que recibe un impulso. Cuando llega a 11, repite este valor y luego comienza a restar 1 hasta llegar a 00, valor que también repite para luego comenzar a sumar 1, repitiéndose el ciclo hasta que se ponga 0 en el registro de control de tono.

Si tomamos los dos bits del último contador y los utilizamos como entrada de un conversor analógico y digital, conseguiremos así una señal que varía periódicamente en el tiempo con la frecuencia dada por la siguiente fórmula:

$$f = \frac{fr}{16*tp}$$

donde:

f == frencuencia de la señal de salida
fr == frecuencua de reloj (en los MSX , 1.7897725 MHz)
tp == valor contenido en el registro de control de tono. En el canal A, sería R0 + 256 * (valor contenido en los bits 0 a 3 en R1).

Asi pues, para conseguir una frecuencia de, aproximadamente 440 Hz (la central del piano) en un MSX, habrá que poner en el registro de control un valor de 1789772.5/(16*440) = 254, es decir, poner en R0 el valor 254 y en R1 0.

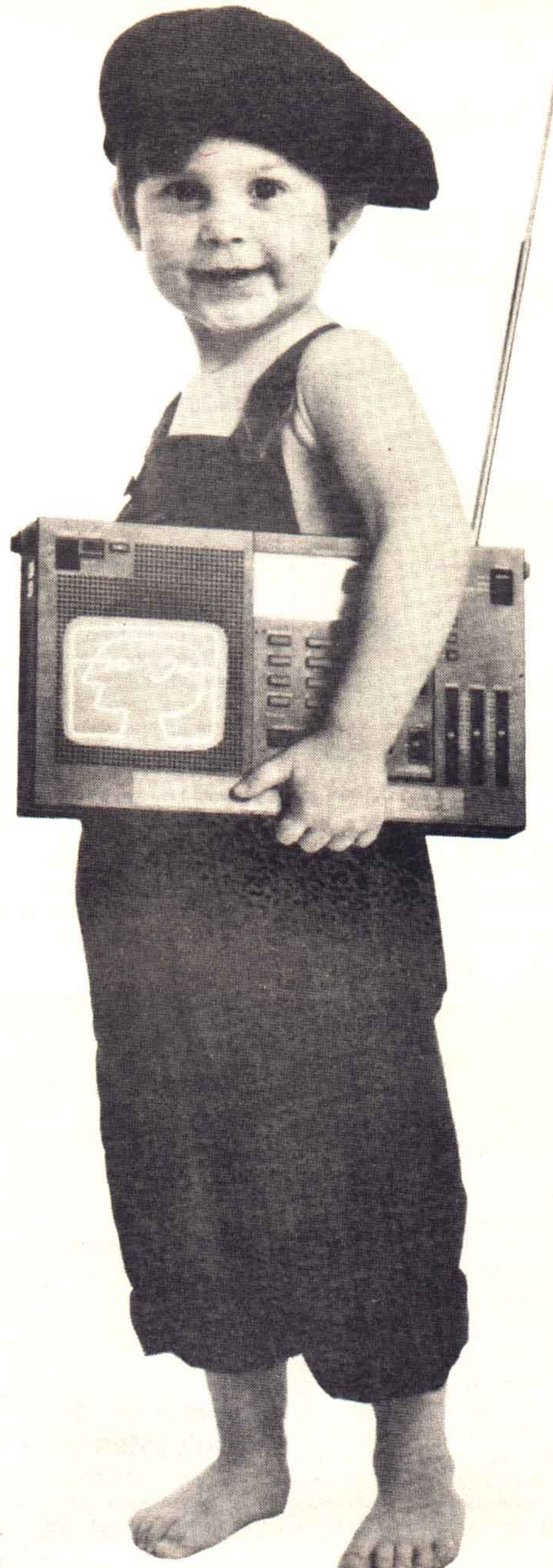
En la figura 3 podemos ver el artículo descrito en esta sección, para aquellos que sepan algo de electrónica y tengan ganas de pensar un poco.

Generación de ruido

La generación de ruidos se realiza por un método similar, sólo que ahora el valor inicial que toma el contador es el que hay en el registro R6, en los bits 0 ó 4, poniéndose ceros en los siete bits superiores. Con la señal que así se genera, se modela la amplitud de un generador de ruido blanco, lo que tiene un efecto realmente ruidoso. Hay que recordar un detalle importante: el canal de ruido es único, y lo que hacemos es decir (mediante R7, como veremos después) en qué canales queremos oírlo.

Generación de envolventes

La generación de envolventes de volumen se controla mediante dos registros diferentes para conseguir



los efectos que podemos ver en la figura 4. La forma la controla el registro R13 y la duración de la rampa los registros R11 y R12.

El registro R13 de forma de envolvente sólo utiliza los 4 bits de menos peso, llamados *CONT*, *ATT*, *ALT* y *HOLD*, cuyas funciones son:

HOLD (Mantener): si está puesto a nivel lógico «1», al acabar el primer ciclo de variación de la envolvente, mantendrá el valor con el que se acabe la rampa (volumen máximo en caso de rampa ascendente o silencio en caso de rampa descendente). En caso de valer «0» la envolvente se repetirá cíclicamente.

ALT. (Alternancia): en caso de que valga «1», se alternan rampas ascendentes y descendentes.

El sonido se genera mediante una serie de valores que se almacenan en unos registros determinados.

ATT. (Ataque): si vale «1», la envolvente comenzará con una rampa ascendente, en caso contrario empezará con una rampa descendente.

CONT. (Continuación): si vale «1», la forma de la envolvente será la definida por el bit *HOLD*. Si no, después del primer ciclo se volverá al estado de silencio.

La duración de la primera rampa (y de las siguientes, si las hubiere) viene definida por los registros R11 y R12, según la fórmula:

$$T = 256\ \frac{256^{*}\ R12 + R11}{fr}$$

Donde fr es la frecuencia de reloj.

Así, para conseguir que un sonido crezca desde el silencio hasta el volumen máximo durante 1 segundo, y luego calle por completo, habrá que poner el valor 4 en R13, el valor 27 en R12 y 79 en R11.

El sistema utilizado para generar fichas envolventes es muy similar al de generación de tonos con algunas diferentes:

a) El primer contador es de 16 bits en vez de 12.

b) El reloj que se introduce en el primer contacto no se divide previamente por 2.

c)El segundo contador es de 4 bits, lo que genera 16 niveles distintos, y es mucho más complicado para poder ser controlado por los bits, *HOD, ALT, ATT* y *CONT*.

d) El convertidor de digital a analógico no es lineal, como el de los tonos, sino exponencial para contrarrestar la respuesta logarítmica de nuestro oído, según se ve en la figura 5.

Para que un canal quede afectado por el generador de envolvente, habremos de poner el bit 4 de su registro (R8 para el canal A, R9 para el canal B y R10 para el canal C) de control envolvente a «a». Caso contrario, su volumen será el fijado por los bits 0 ó 3 de dicho registro, que sustituirán a la entrada del convertidor D/A de envolvente a los 4 bits del segundo contador.

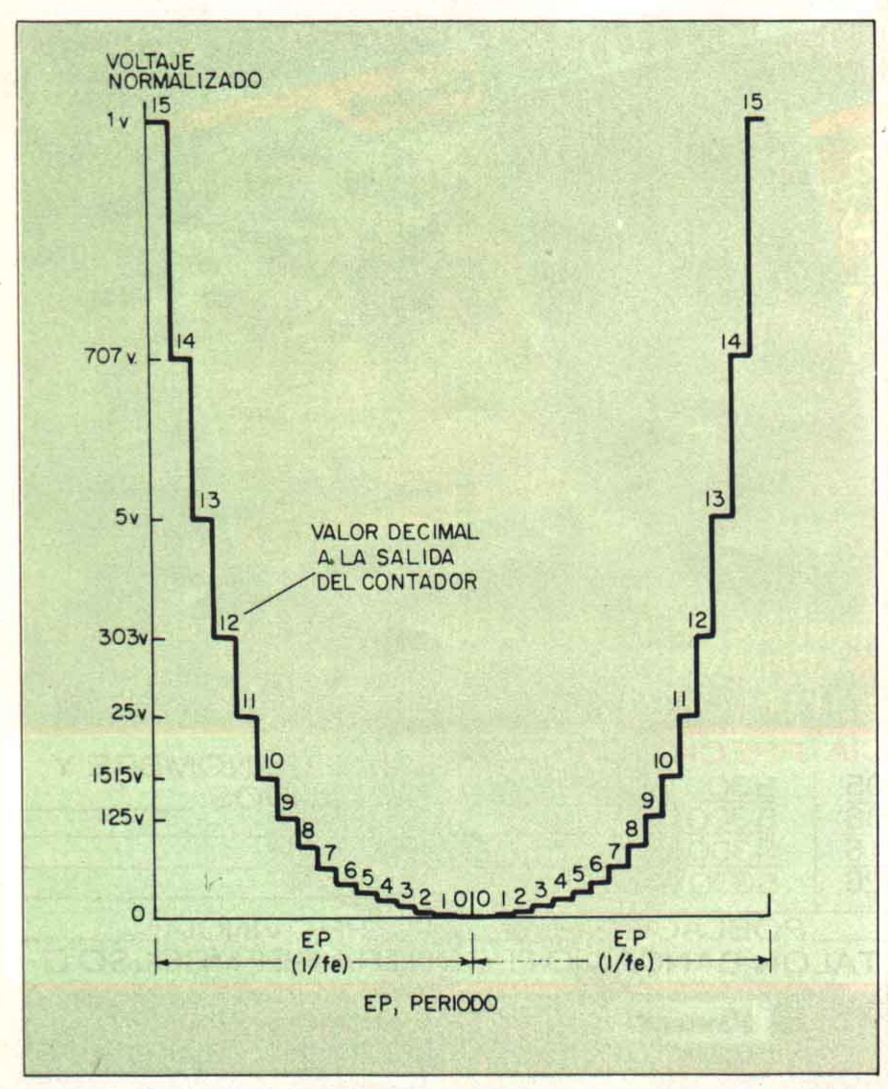


Figura 5a. Envolvente de volúmen generado por el PSG con R15 igual a 10.

Mezcla final de sonidos

Hasta ahora hemos visto como se genera cada parte del sonido. Ahora nuestro problema es entender cómo se unen dichas partes para crear el sonido definitivo que nosotros queremos oír.

Analicemos pues la naturaleza de lo que queremos oír. Nuestra idea principal es percibir un tono base. A este tono podemos añadirle un ruido, lo que significa SUMARLE un ruido. Cuando hablamos de volumen, nos referimos a un valor máximo que pueden alcanzar los picos de nuestra señal de tono, es decir, se trata de una MULTIPLICACION del tono más el ruido por la envolvente de volumen (o el volumen fijo). Todo esto queda claro en la figura 6, donde se han representado los tres canales con algunos bits de control par remarcar la importancia de tener UN SOLO GENERADOR DE RUIDO y UN SOLO GENERADOR DE ENVOLVENTES.

Una serie de registros permiten controlar todas las funciones necesarias de sonido.

Los puertos de entrada y salida

Ya antes habíamos habla de la existencia de un ente asbtracto que habíamos denominado «puertos de entrada y salida (e/s)». para comprender su función, hemos de estu-



diar algo de filosofía informática.

Centrándonos en el Z-80, que es el microprocesador que llevan los MSX, hemos de resaltar que tiene dos espacios de direcciones diferenciados. Uno, que es lo que funciona más rápido, se le denomina clásicamente el espacio de MEMORIA. A él se accede por medio de los 16 bits del bus de direcciones y por la señal de control *MREQ*. Es, pues, un espacio de 65536 lugares posibles. El otro espacio, al que se accede por medio de sólo 8 bits del bus de direcciones y de la señal de control *IOREQ*, es el llamado espacio de E/S.

Es en este segundo espacio donde se sitúan los periféricos. Pero existen varios problemas graves: ni todos los periféricos entienden lo que son las señales de control, ni todos se pueden conectar a un bus de datos, ni todos van tan rápido como el procesador, ni todos entienden las informaciones en binario. Para poder dialogar con este tipo de periféricos se inventaron los puertos e/s, que básicamente son lugares donde se deja un dato hasta que pueda recogerlo el periférico o hasta que se convierta a un formato por él compensible, o donde se recoge un dato cuando el periférico haya podido elaborarlo. En si mismos son memorias de pequeño tamaño (1 octeto, 1/2 octeto) que están conectadas a dos buses, el bus de datos del microprocesador y el bus del periférico, llamado normalmente «bus externo».

Realizada la disquisición filosófica, pasemos a ver el funcionamiento de los puertos del AY-3-8910.

Hay dos puertos en este *chip*, denominado A y B. Los datos intercambiados con el periférico se leen y escriben en los registros R14 (puerto A) y R15 (puerto B). Estos registros sólo pueden funcionar de dos modos: o sólo entrada o sólo salida. El hecho de que funcionen en uno u otro modo viene determinado por unos bits del registro de control (R7) que discutiremos en la próxima sección. Los usos que sugiere el fabricante para estos puertos son muy diversos. Uno de ellos es el de conectar una memoria *ROM* de 256 octetos de tal modo que el puerto A funcione en modo «salida» para direccionar los octetos y el puerto B funcione en modo «entrada» para recoger el dato. La utilidad clásica de esta configuración es la de escribir en dicha *ROM* los

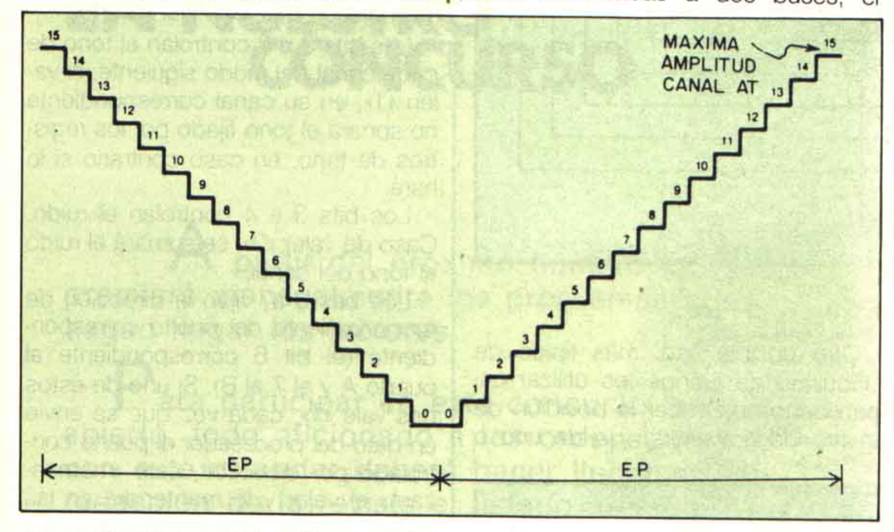


Figura 5b . Respuesta del oído a los distintos volúmenes.

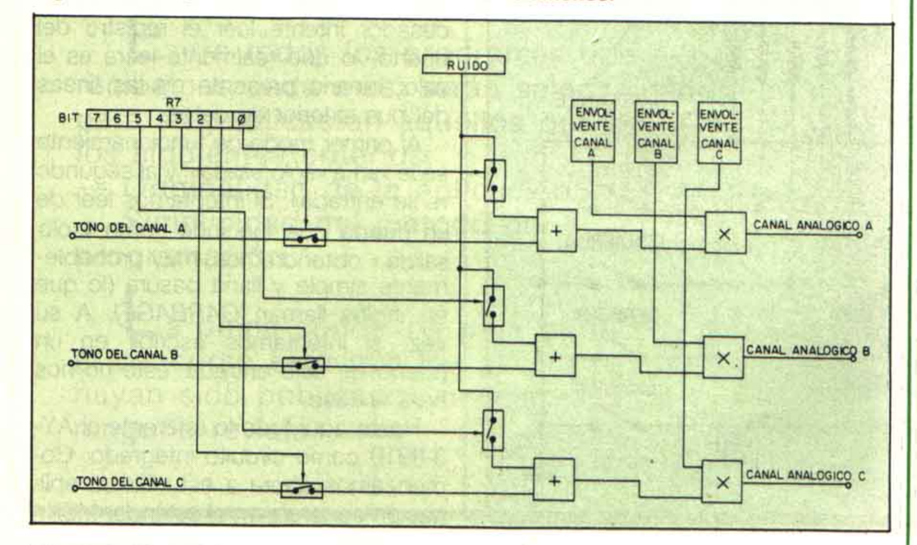


Figura 6. Etapa final del mezclador de ruido, tono y envolvente.

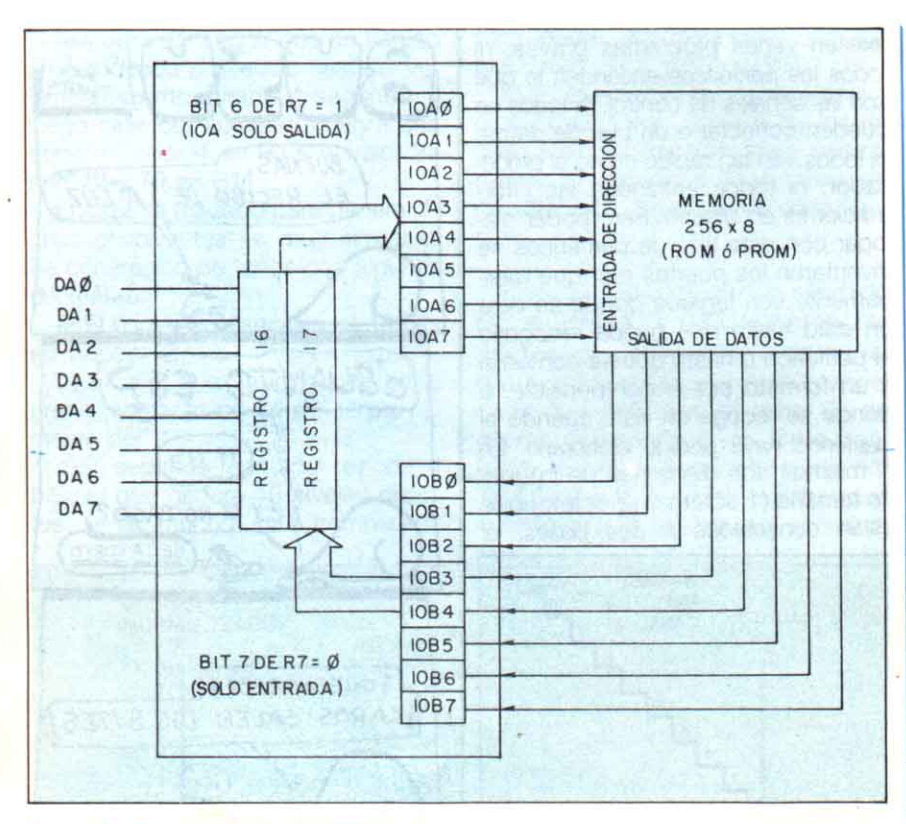


Figura 7a. Disposición con la ROM externa.

parámetros de varios sonidos o efectos especiales para ahorrar el espacio de memoria en la *ROM* principal del procesador.

Otro posible uso, más típico de máquinas de juegos, es utilizar estos puertos para leer la posición de un mando *(joysticks)* y/o de uno o varios botones de disparo o selección, que es el uso principal que se le dá en la familia MSX.

Ambas configuraciones podemos verlas en la figura 7.

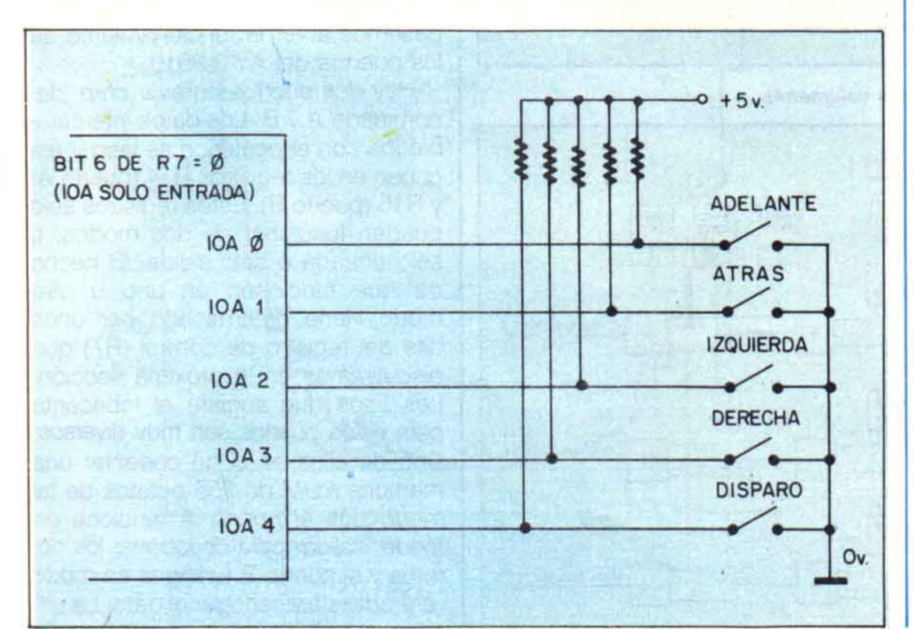


Figura 7b. Disposición con joystick.

El registro general de control R7

El registro R7 cumple la función de controlar la mezcla de ruidos y tonos que se realiza, además de fijar la dirección de funcionamiento de los puertos e/s. Para ello está dividido en tres bloques según se ve en la figura 8: TONO, RUIDO y ENTRADA/SALIDA; los dos primeros formados por tres bits (uno por canal) y el último por dos bits, uno por puerto.

Los bits 0 a 2 controlan el tono de cada canal del modo siguiente: si valen «1», en su canal correspondiente no sonará el tono fijado por los registros de tono, en caso contrario sí lo hará.

Los bits 3 a 4 controlan el ruido. Caso de valer «0», se sumará el ruido al tono del canal.

Los bits 6 a7 fijan la dirección de funcionamiento del puerto correspondiente (el bit 6 correspondiente al puerto A y el 7 al B). Si uno de estos bits vale «1», cada vez que se envíe un dato del procesador al puerto controlado por dicho bit, éste «memorizará» el valor y lo mantendrá en las líneas del bus exterior de datos. En caso contrario, cada vez que el procesador intente leer el registro del puerto, lo que realmente leerá es el valor binario presente en las líneas del bus exterior de datos.

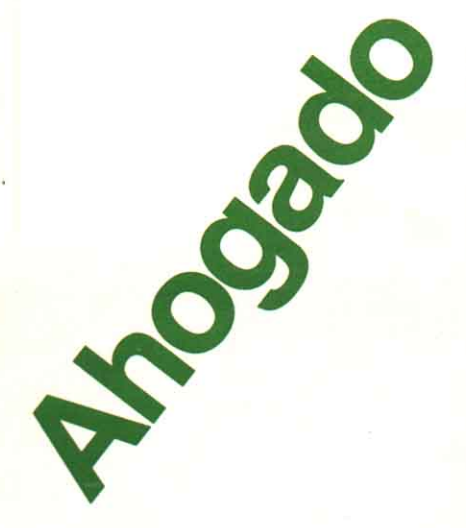
Al primer modo de funcionamiento se le llama «sólo salida», y al segundo «sólo entrada». Si intentamos leer de un puerto que funcione como «solo salida,» obtendremos, muy probablemente simple y llana basura (lo que en inglés llaman *GARBAGE)*. A su vez, si intentamos escribir en un puerto de sólo entrada, éste no nos hará ni caso.

Hasta aquí todo lo referente al AY-3-8910 como circuito integrado. Comenzamos ahora a estudiar su aplicación concreta en el estándar MSX.

Juan Zarategui



GANE
7.000 PTAS.
todos los meses
PARTICIPANDO
EN NUESTRO
CONCURSO
A partir del próximo número MSX premiará mensualmente los programas que hagan llegar los lectores.
Para participar en este concurso abierto, todo aficionado a los ordenadores con este estándar deberá hacer llegar a la redacción de la revista el listado, un cassette y un texto explicativo.
Entre todos los programas que recibamos cada mes, serán seleccionados para su publicación aquellos que reúnan los siguientes criterios:
— Originalidad de la aplicación.
— Simplicidad del método de programación.
La única condición para participar en el concurso será que los programas no hayan sido publicados previamente en ninguna revista.
BANCO DE ESPAÑA
2E1554046
MIL pesetas
1000
CINCO MIL pesetas
5000
14457150



La mayoría de los lectores conocerán el famoso juego del ahorcado, basado en adivinar una palabra en función a las letras que ibamos colocando.

Esta versión del popular juego es muy útil para entretener a los niños y al mismo tiempo hacer que repasen la ortografía de palabras castellanas e inglesas. Consiste en divinar la palabra elegida por el ordenador en uno de los dos juegos, castellanos o inglés. En el primer caso, la palabra se tiene que adivinar dando al ordenador las letras en su correcto orden. Si te equivocas, el ordenador da una pista indicando si la letra está en posición más alta o más baja en el alfabeto. En el segundo caso el orden no influye para adivinar la letra, es decir, las letras se van diciendo y el ordenador las va colocando.

En los dos casos tenemos diez posibilidades de error. Cada vez que nos equivoquemos subirá la marea hasta cubrir al hombrecito que aparece en la pantalla.

La segunda opción es muy sencilla y útil para todo aquel que desée aprender inglés, ya que las palabras a deletrear son fáciles.

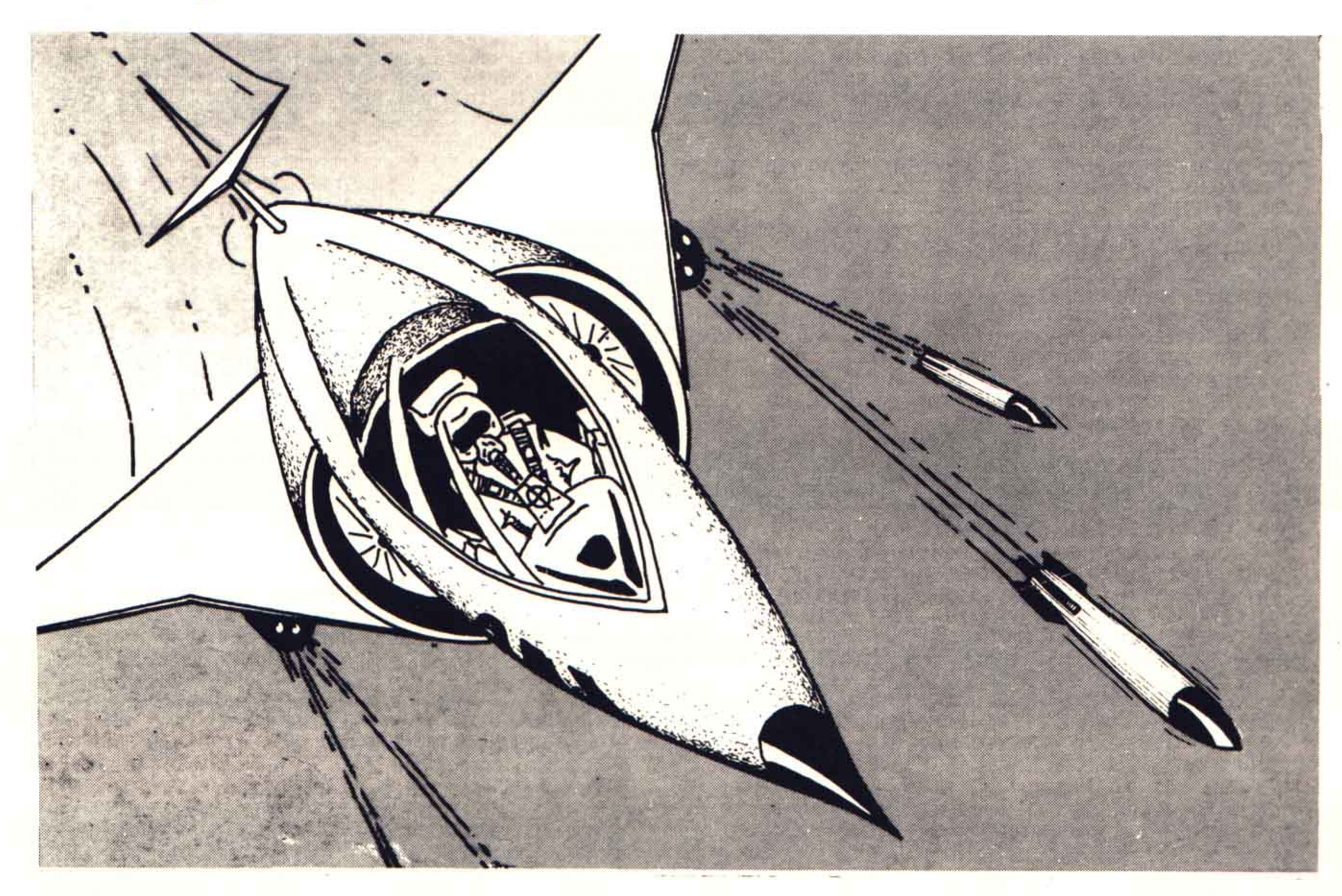
Las líneas 220 y 230 contienen el vocabulario castellano e inglés que se usa en el programa. Los lectores podrán alterarlo, complicarlo y mejorarlo notablemente, cambiando las líneas pertinentes.

Umberto Rinaldi
Huesca

```
10 REM ***A H O G A D O***
20 DIM A$(60):DIM S$(15)
30 KEY OFF:SCREEN 1:COLOR 1,8,4
40 G=0:S$=""
50 LOCATE 7,10:PRINT "A H O G A D O"
60 LOCATE 5,13:PRINT "de  UMBERTO RINALDI"
70 FOR I=1 TO 3000:NEXT
80 CLS:PRINT CHR$(11)
90 ON KEY GOSUB 340,700,980,1040,1040
100 REM **ELECCION DEL JUEGO**
110 PRINT "ELIGE EL JUEGO:"
120 PRINT:PRINT "F1=TIENES QUE ADIVINAR LA PA-    LABRA PONIENDO LAS LETRAS      EN
 ORDEN DE POSICION."
130 PRINT:PRINT "F2=TIENES QUE ADIVINAR SIN       TENER EN CUENTA LA POSI-       CI
ON DE LA LETRA."
140 PRINT:PRINT "F3=FIN DEL JUEGO."
150 KEY(1) ON:KEY(2) ON:KEY(3) ON:KEY(4) ON:KEY(5) ON
160 GOTO 160
170 REM **COLOCACION DEL JUEGO**
180 COLOR 15,4,4
190 CLS:PRINT CHR$(11)
200 INPUT "QUIERES JUGAR EN ESPAÑOL(1), O EN INGLES(2)";B
210 IF B<>1 AND B<>2 THEN PRINT "TE HAS EQUIVOCADO. PRUEVA OTRA VEZ.":FOR I=1 TO
 1000:NEXT I:GOTO 190
220 DATA HOMBRE,COCHE.BARCO,AGUDO,AHORA,ALMOHADA.VENTANA,VACACION,VAGABUNDO,PAJA
RO,HORMIGA,BALLENA,TENEDOR,TIJERAS,SUBMARINO,NIÑO,NIÑA,MUJER,ALBAÑIL,MESITA,ZANA
HORIA,ZARZUELA,CASTELLANO,LLENAR,LLEVAR,LECHE,LUCHA,INTERPONER,INTERPRETAR,INTER
VENCION
230 DATA RUN,SHIFT,SCREEN,SELECT,RETURN,SPACE,YES,NOT,STOP,GO,GREEN,THEN,READ,NE
XT,FOR,DATA,RIGHT,LEFT,WIDTH,OPEN,NAME.MOTOR,HOME,MAN,YELLOW,COMPUTER,WOMEN,CIRC
LE,LINE,PRINT
240 RESTORE
250 FOR I=1 TO 60
260 READ A$(I)
270 NEXT I
280 R=RND(-TIME)
290 X=INT(RND(1)*30+1)
300 IF B=2 THEN X=X+30
310 N=1
320 IF A=2 THEN RETURN 730
330 RETURN 370
340 REM **SUBRUTINA DEL 1 JUEGO**
350 A=1
360 GOSUB 170
370 CLS
380 LOCATE 1,190:PRINT S$:PRINT CHR$(11)
390 PRINT SPC(10);"1 JUEGO":PRINT
400 INPUT "DIME UNA LETRA";B$
410 L=LEN(A$(X))
420 IF B$<>MID$(A$(X),N,1) THEN 490
430 N=N+1:S$=S$+B$
440 PRINT "BRAVO!"
450 PLAY "V8BGV15BGV8BGV3BG"
460 FOR I=1 TO 3000:NEXT
470 IF N>L THEN 990
480 GOTO 370
490 REM ***SUBRUTINA DEL DIBUJO***
```

```
500 G=G+1
510 IF G=10 GOTO 620
520 CLS:PRINT CHR$(11)
530 PRINT "TE HAS EQUIVOCADO.":PRINT "TEN CUIDADO! TE ESTAS AHOGAN DO!"
540 ON A GOTO 550,570
550 IF ASC(B$)<ASC(MID$(A$(X),N,1)) THEN PRINT "ES UNA LETRA MAS ALTA."
560 IF ASC(B$)>ASC(MID$(A$(X),N,1)) THEN PRINT "ES UNA LETRA MAS BAJA."
570 PRINT:PRINT:PRINT TAB(15) "@":PRINT TAB(14) "/n\":PRINT TAB(14) "J L"
580 PRINT CHR$(11):PRINT:PRINT
590 FOR I=G TO 10:PRINT:NEXT I
600 PRINT TAB(4) "_/\_"
610 PRINT "^^^^^\__/^^^^^^^^^^^^^^^^^^^^^^^^^^^^"
620 FOR I=1 TO 4
630 PRINT "^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^":NEXT I
640 IF G=10 GOTO 680
650 FOR I=1 TO 3000:NEXT
660 IF A=1 THEN 370
670 IF A=2 THEN 750
680 PRINT:PRINT:PRINT "TE HAS AHOGADO."
690 GOTO 1000
700 REM ***SUBRUTINA DEL 2 JUEGO***
710 A=2
720 GOSUB 170
730 L=LEN(A$(X))
740 FOR I=1 TO L:S$(I)=CHR$(46):NEXT I
750 CLS:Z=0:LOCATE 1,190
760 FOR I=1 TO L:PRINT S$(I);:NEXT I
770 PRINT CHR$(11)
780 PRINT SPC(10);"2 JUEGO":PRINT
790 INPUT " DIME UNA LETRA";B$
800 PRINT
810 FOR I=1 TO L
820 IF B$=MID$(A$(X),I,1) THEN S$(I)=B$:Z=Z+1
830 NEXT I
840 FOR I=1 TO L
850 IF S$(I)=CHR$(46) THEN 880
860 NEXT I
870 GOTO 990
880 IF Z=0 THEN 490
890 IF Z>0 THEN PRINT "BRAVO!"
900 PLAY "V8BGV15BGV8BGV3BG"
910 LOCATE 1,190
920 FOR I=1 TO L
930 PRINT S$(I);
940 NEXT I
950 FOR I=1 TO 3000:NEXT
970 GOTO 750
980 PRINT:PRINT:PRINT:PRINT "FIN":END
990 PRINT "HAS ACERTADO."
1000 PRINT "LA PALABRA ES:"
1005 LOCATE 1,190:PRINT A$(X)
1010 FOR I=1 TO 3:PLAY "V8BGBV15DGB":NEXT
1020 FOR I=1 TO 6000:NEXT I
1030 GOTO 30
1040 CLS:PRINT CHR$(11)
1050 PRINT "HAS DADO EN LA TECLA EQUIVO- CADA.INTENTALO OTRA VEZ."
1060 FOR I=1 TO 3000:NEXT I
1070 GOTO 80
```

Ataque



Los marcianos se han hecho con el control de la Tierra, pero tu has escapado de ellos con una simple nave y estás a punto de llegar al Cuartel General enemigo. Mata a los guardianes y luego esquiva las naves y meteoritos que protegen el Gran Pasillo, pero ¡cuidado! Ellos estan programados para estrellarse contra tí, por lo que tendrás que evitarlos. Si consigues aguantar hasta el final vencerás, de lo contrario...

Para guiar la nave, utiliza las teclas del cursor (izquierda/derecha) y para disparar, pulsa la barra espaciadora. Es un juego simple, en concepción, pero no tan fácil de jugar. Hay que exterminar las naves enemigas que aparecerán al comienzo del juego. Son seis, e irán apareciendo en lugares aleatorios. De distintos colores, cualidad que a veces no preocupa, pero que en este caso si, puesto que el que posea televisión en blanco y negro, se las verá y deseará para aniquilar la última nave, ya que se vuelve invisible a los ojos, lo que complicará y hará más interesante el juego.

La segunda pantalla, también tiene su grado de dificultad, pero quizás no llegue a la complicación de la primera. En esta, tienes que tratar de esquivar naves y meteoritos, como se explicó al principio.

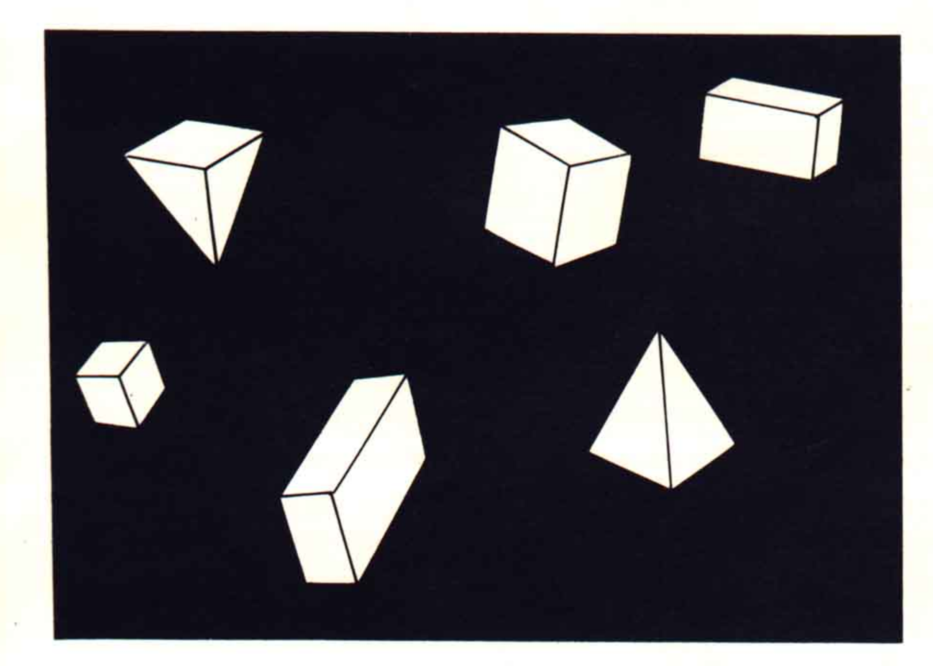
En la parte superior de la pantalla, aparece un reloj que tendrás que vigilar, ya que te indicará el tiempo que te queda para terminar la misión. ¡Suerte!

```
10  Z=5:ST=0:GOSUB950:DEFUSR=&H156:OPEN"grp:"FOROUTPUTAS#1
20 COLOR15,4,7:SCREEN0,,0:KEYOFF
30 PRINTSPC(10)" ┌──────┐ ":PR=0:SC=0
40 PRINTSPC(10)" |ATAQUE| ":CX=CX+1
50 PRINTSPC(10)" └──────┘ "
60 PRINT:PRINT:PRINT:PRINT"                              "
70 PRINT:PRINT:PRINT:PRINT"  -> :  Derecha":PRINT:PRINT" <- :  Izquierda":PRINT
:PRINT" Esp.:  Disparo":PRINT:PRINT" Tab :  Record":PRINT:PRINT" ESC :  Menu"
80 LOCATE3,22:PRINT"Pulsa el ESPACIO para jugar":X=USR(X)
90 X$=INKEY$:IFX$=CHR$(9)THENGOSUB860:GOTO90ELSEIFX$<>" "THEN90
100 BEEP:PLAY"M3000S9","M3000S9":PLAY"O3T255L30AB","O3T255L30AB"
110 KEY(1)OFF:COLOR15,14,0:SCREEN2,2:GOSUB960:TM=148
120 LINE(0,0)-(256,49),1,BF
```

```
130 FORT=0TO100:X=INT(RND(1)*256):Y=INT(RND(1)*49):PSET(X,Y),10:NEXT:ST=0
140 A=.1:D=40:C=0:SC=0
150 X=128:HT=160:XG=INT(RND(1)*TIME):XG=INT(RND(1)*250)+10:YG=INT(RND(1)*100)+40
160 FORT=96TO70STEP-1:LINE(0,T/A)-(256,T/A):A=A+.05:NEXT
170 W=16.2:H=.1:IN=4
180 FORX=95TO36STEP-5:LINE(X,50)-(0,W/H):H=H+.02:NEXT
190 ONINTERVAL=4GOSUB660
200 SPRITEON
210 FORX=100TO146STEP5:LINE(X,50)-(X-140+D,192):D=D+20:NEXT
220 LINE(X,50)-(256,192)
230 Y=19.2:S=.1
240 FORX=150TO224STEP5:LINE(X,50)-(256,Y/S):S=S+.02:NEXT:GOSUB380
250 TIME=0:G=0:ONSPRITEGOSUB450:X=USR(X)
260 GOSUB320:IFTIME>900THENPR=1:INTERVALOFF:SPRITEOFF:SOUND8,0:GOTO860ELSEP=TIME
270 IFSTICK(0)=3ANDX<250-ZTHENX=X+Z
280 IFSTICK(0)=7AND X>ZTHEN X=X-Z
290 IFSTRIG(0)=-1THENGOSUB330
300 IFINKEY$=CHR$(27)THEN20
310 GOSUB690:T=TIME:GOSUB390:GOTO260
320 SPRITEOFF:PUTSPRITE0,(X,176),4,0:PUTSPRITE1,(X,171),9,1:SPRITEON:RETURN
330 IFST=1THENRETURNELSEST=1:LT=X:GOSUB1170:INTERVALON:RETURN
340 SCREEN0:LOCATE12,12:PRINT"┌─────────┐"
350 LOCATE12,13:PRINT"|VENCISTE|"
360 LOCATE12,14:PRINT"└─────────┘"
370 FORT=0TO1000:NEXT:SCREEN0:COLOR15,4,7:P=400:GOSUB480:GOTO110
380 LINE(58.18)-(10,10),1,BF:PRINT#1,"Tiempo":LINE(58,10)-(TM,18),5,BF:RETURN
390 LINE(TM-T/10,10)-(TM,18),1,BF:RETURN
400 INTERVALOFF:SPRITEOFF:PUTSPRITE0,(X,175),3,30:PUTSPRITE1,(X,174),12,30:GOSUB
440:FORT=0TO50STEP3:CIRCLE(X+8,182),T,1:NEXT
410 FORT=0TO1000:NEXT:GOTO860
420 IFGH=114THENINTERVALOFF:SPRITEOFF:GOTO340
430 IFINKEY$=CHR$(27)THENINTERVALOFF:SPRITEOFF:GOTO20ELSEGOTO610
440 SOUND0,0:SOUND6,15:SOUND7,7:SOUND12,100:FOR T=8 TO 10:SOUNDT,16:NEXT:SOUND13
,0:RETURN
450 REM
460 INTERVALOFF:SPRITEOFF:PUTSPRITE4,(XG,YG),6,3:PUTSPRITE5,(XG,YG+8),9,3:PUTSPR
ITE2,(LT,HT),,30:GOSUB440:FORT=0TO50:NEXT:PUTSPRITE5,(XG,YG+13),,30:PUTSPRITE4,(
XG,YG),,30:PUTSPRITE2,(LT,HT),,30:SC=SC+100:HT=160:ST=0:IFG<6THENG=G+1:RETURNELS
EGOTO520
470 P=TIME:SCREEN0:COLOR 15,4,7
480 LOCATE12,12:PRINT"┌──────────┐"
490 LOCATE12,13:PRINT"|Bonus"+STR$(900-P):LOCATE23,13:PRINT"|"
500 LOCATE12,14:PRINT"└──────────┘"
510 FORT=0TO1000:NEXT:SC=SC+900-P:RETURN
520 GOSUB470:SCREEN2,2
530 LINE(0,191)-(127,64):LINE(256,191)-(128,64):L=127:Z=1:H=0:T=64:CX=2
540 LINE(L,1)-(L,T-CX):LINE(L,T-CX)-(0,T-CX):LINE(L,T)-(L+Z,T):IFH/2<>INT(H/2)AN
DH>2THENPAINT(L+10,T-1),15
550 LINE(L+Z,T)-(L+Z,T-CX):LINE(L+Z,T-CX)-(256,T-CX):H=H+1:Z=Z+2*H:L=L-H:CX=CX+H
*.5
560 T=T+H:IFT<192THEN540
570 LINE(127,62)-(128,64),4,BF:LINE(0,0)-(256,61),1,BF
580 FORT=0TO100:PSET(INT(RND(1)*250)+10,INT(RND(1)*61)),10:NEXT:YM(2)=35:YM(3)=2
5
590 LINE(0,125)-(125,62.5):LINE(256,125)-(130,62.5):GOSUB1160:X=100:Z=5:FX=26.5:
YM(0)=400:YM(1)=45:SF(1)=1:SF(0)=1:SF(2)=1:SF(3)=1:TM=170:GOSUB380:GH=0:ONINTERV
AL=8GOSUB630:INTERVALON
600 ONSPRITEGOSUB400:SPRITEON
610 SP=0:GH=GH+2:T=GH:GOSUB390
620 GOSUB670:SC=SC+1:GOTO420
630 IFSTICK(0)=3ANDX<238-ZTHENX=X+Z:SP=4
640 IFSTICK(0)=7ANDX>ZTHENX=X-Z:SP=2
650 SPRITEOFF:PUTSPRITE0,(X,175),3,SP:PUTSPRITE1,(X,174),12,SP+1:FORT=0TO10:NEXT
:SPRITEON:RETURN
660 IFHT<10THENPUTSPRITE2,(LT,HT),,30:HT=160:ST=0:INTERVALOFF:RETURNELSEHT=HT-15
:PUTSPRITE2,(LT,HT),10,2:RETURN
670 SR=SR+1:IFSR=4THENSR=0
680 GOSUB800:RETURN
690 SPRITEOFF:PUTSPRITE4,(XG,YG),7-G,4:PUTSPRITE5,(XG,YG+13),6-G,5:GOSUB700:RETU
RN
700 ZB=YG:ONINT(RND(1)*7)GOSUB720,730,740,750,760,770,780,790:SPRITEON
```

```
710 IFYG<10ORYG>100THENYG=ZB:RETURNELSERETURN
720 XG=XG-Z:RETURN
730 XG=XG+Z:RETURN
740 YG=YG-Z:RETURN
750 YG=YG+Z:RETURN
760 XG=XG-Z:YG=YG-Z:RETURN
770 XG=XG+Z:YG=YG-Z:RETURN
780 XG=XG+Z:YG=YG+Z:RETURN
790 XG=XG-Z:YG=YG+Z:RETURN
800 IFYM(SR)>189THENYM(SR)=55:XM(SR)=123:V(SR)=6:U(SR)=6:GOSUB850
810 IFYM(SR)<54THENYM(SR)=YM(SR)+1:RETURNELSEIFYM(SR)=54THENYM(SR)=400:GOTO800
820 IFXM(SR)<XTHENDR=1ELSEIFXM(SR)>XTHENDR=-1ELSEDR=0
830 XM(SR)=XM(SR)+K(SR)+DR*V(SR):YM(SR)=YM(SR)+V(SR):V(SR)=V(SR)/.9609:U(SR)=U(S
R)+.354:K(SR)=K(SR)/.9609
840 PUTSPRITE25+SR,(XM(SR),YM(SR)),6,U(SR):RETURN
850 K(SR)=INT(RND(1)*12)-6:RETURN
860 COLOR 15,4,7:SCREEN0:PRINTSPC(11)"┌────────┐"
870 PRINTSPC(11)"| RECORD |":PRINTSPC(11)"└────────┘"
880 IF SC<SC(7)THEN930
890 FORT=1TO7:IFSC>HS(T)THEN910
900 NEXT:GOTO930
910 FORL=7TOTSTEP-1:HS$(L+1)=HS$(L):HS(L+1)=HS(L):NEXT
920 HS(T)=SC:LOCATE,22:X=USR(X):INPUT"¿Como te llamas";N$:N$=MID$(N$,1,15):HS$(T
)=N$
930 LOCATE,4:FORT=1TO7:PRINT"_";T;"_";HS(T),HS$(T):PRINT:NEXT:FORT=0TO250:IFINKE
Y$=CHR$(27)THENT=249:PR=1:NEXTELSENEXT
940 IFPR=0THENRETURNELSESCREEN0,,0:GOTO20
950 FORT=1TO7:HS$(T)=" MSX Magazine":HS(T)=100:NEXT:RETURN
960 RESTORE
970 A$="":FORT=0TO5:FORG=0TO31:READA:A$=A$+CHR$(A):NEXT:SPRITE$(T)=A$:A$="":NEXT
:RETURN
980 DATA0,0,0,16,17,57,59,63,63,63,61,57,57,57,56,16,0,0,0,4,68,78,110,126,126,1
26,94,78,78,78,14,4
990 DATA0,0,0,0,1,3,7,6,4,0,0,0,0,0,0,0,0,0,0,64,224,240,176,144,128,128,128,1
28,128,128,128
1000 DATA0,0,0,0,0,0,0,0,0,0,0,0,0,0,64,64,64,64,64,192,192,192,192,192,128,
128,128,128,128,128
1010 DATA16,0,40,4,0,71,39,15,31,30,15,7,68,8,16,0,0,0,8,32,16,40,194,228,104,19
7,160,16,8,32,2,0
1020 DATA0,0,0,0,0,0,0,0,0,0,24,24,28,28,4,0,0,0,0,0,0,0,0,0,0,48,48,112,112
,64
1030 DATA1,3,3,7,7,5,61,55,2,2,0,0,0,0,0,0,0,128,128,192,192,64,120,216,128,128,
0,0,0,0,0,0
1040 DATA0,0,0,0,0,240,33,19,31,22,31,1,0,0,0,0,0,0,0,0,15,132,200,248,104,248
,128,0,0,0,0
1050 DATA0,0,0,1,97,241,3,30,12,0,0,0,0,0,0,0,0,0,128,134,143,192,120,48,0,0,0
,0,0,0,0
1060 DATA0,0,0,0,0,0,1,3,1,19,33,83,151,13,7,2,8,16,32,88,4,14,91,254,252,216,17
6,248,208,128,0,0
1070 DATA0,0,0,0,16,12,15,6,4,6,100,198,140,8,0,0,0,48,96,64,0,24,240,160,0,0,0,
0,0,0,0,0
1080 DATA16,8,4,26,32,112,218,127,63,27,13,31,11,1,0,0,0,0,0,0,0,128,192,128,2
00,132,202,233,176,224,64
1090 DATA0,12,6,2,0,24,15,5,0,0,0,0,0,0,0,0,0,0,8,48,240,96,32,96,38,99,49,1
6,0,0
1100 DATA0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,128,128,0,0,0,0,0,0
1110 DATA0,0,0,0,0,0,3,7,1,0,0,0,0,0,0,0,0,0,0,0,128,192,192,128,0,0,0,0,0,0
1120 DATA0,0,0,0,0,7,2,15,3,1,0,0,0,0,0,0,0,0,0,0,192,224,64,224,224,128,0,0,0
,0,0
1130 DATA0,0,0,0,7,3,14,31,13,7,1,3,0,0,0,0,0,0,0,0,192,240,224,64,240,120,48,22
4,192,0,0,0
1140 DATA0,0,0,7,3,14,31,63,29,15,3,1,3,0,0,0,0,0,0,224,236,248,112,224,184,220,
164,216,240,128,0,0
1150 DATA31,63,30,4,31,31,63,249,61,127,31,15,31,23,0,1,240,224,238,252,60,184,2
40,240,252,255,62,124,248,248,192,128
1160 RESTORE1040:FORT=0TO11:FORG=0TO31:READA:D$=D$+CHR$(A):NEXT:SPRITE$(T)=D$:D$
="":NEXT:RETURN
1170 SOUND11,255:SOUND12,10:SOUND13,9:SOUND7,124:SOUND8,0:PLAY"M1000S902A":RETUR
N
1180 REM
```

El juego de las parejas



Muchas veces no utilizamos nuestra capacidad de concentración hasta el límite de sus posibilidades, lo que hace que a veces nuestros reflejos sean lentos.

Para estos casos, lo ideal es contar con un juego que no obligue a pensar, pero si utilizar nuestro pequeño cerebro, pues para algo lo tenemos. En suma, en este juego, nos veremos en la desagradable situación de tener que acordarnos de todas las posiciones en la que van apareciendo las cartas que nos interesen.

Dos jugadores se enfrentarán para ver quien de los dos consigue acertar el máximo número de parejas posibles, lo que no es fácil, teniendo en cuenta que las cartas están boca abajo y en un orden totalmente aleatorio. Uno de los dos indicará al ordenador las coordenadas deseadas, éste dará la vuelta a las cartas y comprobará si éstas son o no son iguales. De ser así el jugador que ha acertado suma puntos. El juego continuará hasta que acabemos con todas las cartas.

Salvador Terrado
Barcelona

```
1 CLS:KEY OFF:LOCATE 8,8:PRINT" *** JUEGO DE PAREJAS ***"
2 PRINT" por Salvador Terrado"
3 LOCATE 1,22:PRINT"pulse cualquier tecla para seguir"
4 IF INKEY$="" THEN 4
10 DIM O1$(36),V$(36),V(36)
20 SCREEN 0
30 Q$="ABCDEFGHIJKLMNOP*+ABCDEFGHIJKLMNOP*+"
40 G=1
50 KEY OFF:CLS
60 COLOR 15,4,4
70 FOR K= 1 TO 36
80 O1$(K)=CHR$(255):NEXT K
90 GOTO 380
100 PSET(185,150):COLOR 15:PRINT#1," barra"
110 IF INKEY$<>" " THEN 110
120 PSET(185,150):COLOR 4:PRINT#1,STRING$(8,"█"):RETURN
130 REM
140 IF O1$(O)="#"THEN PSET(185,150):COLOR 15:PRINT#1,"  o vale":GOTO 670 ELSE RE
TURN
150 IF O1$(O1)="#"THEN PSET(185,150):COLOR 15:PRINT#.," no vale":GOTO 740 ELSE R
ETURN
160 PSET (190,100):COLOR 15:PRINT#1,"v 1.- █"
170 PSET (190,110):COLOR 15:PRINT#1,"v 2.- █"
```

```
180 C$=INKEY$
190 IF C$<"1" OR C$>"6" THEN 180 ELSE W=VAL(C$)
200 PSET(240,100):COLOR 1:PRINT#1,C$
210 CC$=INKEY$
220 IF CC$<"1" OR CC$>"6" THEN 210 ELSE WW=VAL(CC$)
230 PSET(240,110):COLOR 1:PRINT#1,CC$
240 RETURN
250 LOCATE 8,10:PRINT"GENERANDO VALORES"
260 PLAY"abcdefgabcdefg"
270 Z=1:A=0
280 A=A+1
290 V(A)=INT(RND(-TIME)*36+1)
300 IF V(A)>36 THEN 290
310 FOR H=A-1 TO 1 STEP -1
320 IF V(A)=V(H) THEN V(A)=Z:Z=Z+1:GOTO 300
330 NEXT H
340 V=V(A)
350 V$(V)=MID$(Q$,G,1):G=G+1
360 IF A<36 THEN 280
370 RETURN
380 CLS:PRINT TAB(10)"INSTRUCCIONES"
390 PRINT:PRINT:PRINT"El juego consiste en tratar de hacer   parejas.Cada pareja
 equivale a 1 punto excepto los caracteres <*> y <+> que   valen por 5 puntos"
400 PRINT"!!! NO VALE APUNTAR LAS POSICIONES      USA SOLO TU MEMORIA"
410 PRINT:PRINT:PRINT TAB(10)"S U E R T E"
420 LOCATE 2,20:PRINT"pulse la barra de espacio"
430 IF INKEY$<>" " THEN 430
440 CLS:LOCATE 2,5:INPUT"número de jugadores ";N
450 IF N<2 OR N>9 THEN 440
460 FOR J= 1 TO N
470 LOCATE 2,J+5:INPUT" nombre ";N$(J)
480 IF LEN(N$(J))>8 THEN LOCATE 2,J+5:PRINT STRING$(37," "):GOTO 470
490 P(J)=0:NEXT J
500 CLS:GOSUB 250
510 COLOR 15,1,4
520 SCREEN 2
530 A$="D156":B$="R156"
540 FOR X=20 TO 176 STEP 26
550 PSET(X,20):DRAW A$
560 PSET(20,X):DRAW B$:NEXT X
570 OPEN "grp:" AS#1
580 FOR X=25 TO 176 STEP 26:PSET(3,X+3),1:X1=X1+1:PRINT#1,X1
590 PSET (X-2,7),1:PRINT#1,X1:NEXT X
600 J=1
610 PSET(185,30):COLOR 4:PRINT#1,STRING$(8,"█")
620 PSET(185,50):COLOR 15:PRINT#1,"PUNTOS"
630 PSET(185,60):COLOR 4:PRINT#1,"████"
640 PSET(185,30):COLOR 15:PRINT#1," ";N$(J)
650 PSET(185,60):COLOR 15:PRINT#1," ";USING"##";P(J)
660 PSET(185,150):COLOR 4:PRINT#1,STRING$(8,"█")
670 GOSUB 160
680 PSET(185,150):COLOR 4:PRINT#1,STRING$(8,"█")
690 A=W:A1=W*26+4
700 B=WW:B1=WW*26+4
710 O=6*(A-1)+B:GOSUB 140
720 O1$(O)="#"
730 PSET(A1+2,B1):COLOR 15:PRINT#1,V$(O)
740 GOSUB 160
750 PSET(185,150):COLOR 4:PRINT#1,STRING$(8,"█")
760 C=W:C1=W*26+4
770 D=WW:D1=WW*26+4
780 O1=6*(C-1)+D:GOSUB 150
790 O1$(O1)="#"
```

```
800 PSET(C1+2,D1):COLOR 15:PRINT#1,V$(O1)
810 GOSUB 100
820 IF V$(6*(A-1)+B)<>V$(6*(C-1)+D) THEN PSET(A1,B1):COLOR 1:PRINT#1,"█":PSET(C1
,D1):COLOR 1:PRINT#1,"█":O1$(O)=CHR$(255):O1$(O1)=CHR$(255):GOTO 870
830 O1$(O1)="#":O1$(O)="#"
840 IF V$(6*(A-1)+B)="+" OR V$(6*(A-1)+B)="*"THEN P(J)=P(J)+5 ELSE P(J)=P(J)+1
850 M=M+2:IF M=36 THEN PSET(185,150):COLOR 15:PRINT#1," fin":FOR S=1 TO 250:NEXT
 S:GOTO 890
860 GOTO 610
870 J=J+1:IF J>N THEN J=1
880 GOTO 610
890 SCREEN 0
900 FOR X= 1 TO N
910 FOR J= 1 TO N-1
920 IF P(J)<P(J+1)THEN A=P(J+1):P(J)=P(J+1):P(J+1)=A ELSE GOTO 940
930 A$=N$(J+1):N$(J)=N$(J+1):N$(J+1)=A$
940 NEXT J:NEXT X
950 CLS:PRINT TAB(10)"EL GANADOR HA SIDO":PRINT:PRINT
960 PRINT TAB(10) N$(1):PRINT:PRINT:PRINT "CON ";P(1)
970 PRINT"pulse <enter> para finalizar o         <esc> para una nueva partida"
980 Z$=INKEY$
990 IF Z$=CHR$(&H1B) THEN CLOSE#1:GOTO 20
1000 IF Z$=CHR$(&HD) THEN END
1010 GOTO 980
```

SERVICIO DE EJEMPLARES ATRASADOS

ESTOS SON LOS EJEMPLARES DE MSX MAGAZINE APARECIDOS EN EL MERCADO CON UN RESUMEN DE SU CONTENIDO



Núm. 1
¿Qué es el MSX? Su BASIC, periféricos, programas, software.



Núm. 2
Generación de sonido, MSX-DOS, el ordenador por dentro, programas, noticias.



Núm. 3
Los joysticks, 256 caracteres programables, Z80 corazón de león, compro/vendo/cambio.



Núm. 4
Las comunicaciones entre ordenadores, la jerga informática, trucos, rincón del lector.



Núm. 5
Comandos de entrada/salida, el BASIC MSX comparado con Spectrum y Commodore 64, Código Máquina.



Núm. 6
Los 8 magníficos (test gigante), el bus de expansión, los misterios de la grabación, programas.

PARA HACER SU PEDIDO, RELLENE ESTE CUPON, HOY MISMO Y ENVIELO A MSX MAGAZINE BRAVO MURILLO, 377. Tel. 733 96 62 - 28020 MADRID

Ruego me envíen los siguientes números atrasados ....

al precio de **250** ptas. cada uno. Cuyo importe abonaré:

☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO

☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta ....

Fecha de caducidad ....

NOMBRE ....

DIRECCION ....

POBLACION .... C.P. ....

PROVINCIA ....



La pantalla es el medio más importante que posée el ordenador para comunicarse con el programador. Para que esa comunicación sea efectiva, el ordenador requiere una memoria especial que disponga lo que tiene que aparecen en pantalla, y que en todo momento sepa qué es exactamente todo lo que hay dibujado o escrito.

Esa memoria especial es la memoria de vídeo, de la que ya se habló en esta revista (ver: «256 Caracteres Programables», MSX magazine n.° 3), que se divide en 16384 «casillas» u «octetos», numerados del 0 al 16383. En cada octeto se puede introducir un número decimal del 0 al 255, o su equivalente en hexadecimal o en binario. Este número puede ser introducido con la orden *VPOKE* de la siguiente manera:

VPOKE octeto, n.° a introducir.

La memoria de vídeo tiene cuatro sistemas de almacenamiento de datos, uno por cada uno de los cuatro modos *SCREEN*. En el momento en el que se cambia el modo *SCREEN* del ordenador, todo cambia en la manera de almacenar datos de la memoria de vídeo, a excepción de las figuras móviles, de las que ya hablaremos en próximos números. En cada uno de los cuatro sistemas, el ordenador señala en la memoria de vídeo zonas de almacenamiento de datos, conocidas como «tablas». En los ordenadores MSX disponibles, esas tablas están situadas en lugares concretos de la memoria, pero EXISTE LA POSIBILIDAD de que en otros ordenadores MSX esas tablas estén situadas en lugares distintos. ¡Pero no te alarmes pensando que te vamos a contar algo que no tiene nada que ver con tu ordenador! Esas tablas seguro que se hallan en tu ordenador, lo que sucede es que no sabemos dónde, lo cual es muy fácil de averiguar. Si por ejemplo quieres saber dónde se encuentra la tabla número 0 tienes que escribir:

PRINT BASE (0)

y el resultado será el octeto en el que comienza la tabla 0, que en nuestros ordenadores es el octeto número 0.

La integración de las tablas en los sistemas de almacenamiento es muy sencilla: en el sistema de *SCREEN* 0 están las tablas 0 y 2, en *SCREEN* 1 están las tablas 5, 6, 7, 8 y 9, en *SCREEN* 2 están las tablas 10, 11, 12, 13, y 14., y en *SCREEN* 3 están las tablas 15, 16, 17, 18 y 19. No existen tablas numeradas con 1, 3, 4 ó 16.

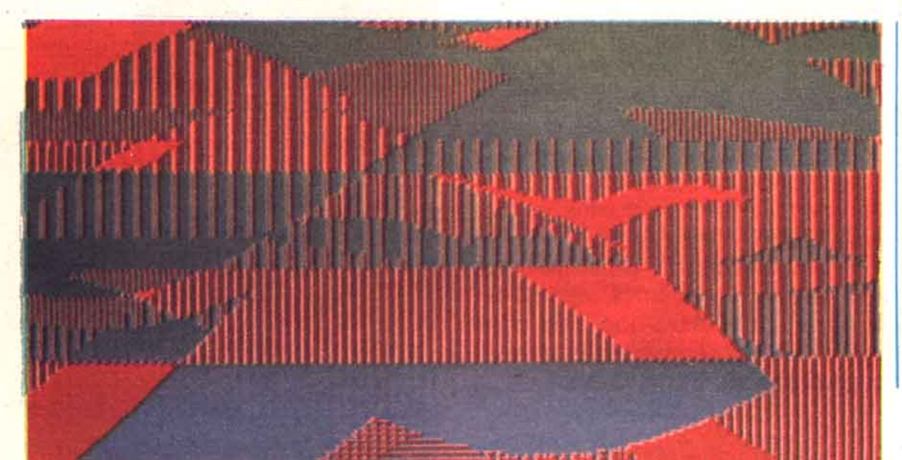
Cada una de las tablas tiene una función y una extensión concreta. Por ejemplo, la tabla 0 ocupa 960 octetos. Si el octeto en el que comienza en tu ordenador, ésta tabla es el octeto número 0, como en los nuestros, tu tabla estará situada entre los octetos 0 y 959. Si en tu ordenador la tabla 0 comienza en el octeto n, estará comprendida entre los octetos n y n + 959, ambos inclusive.

La tabla 0: ¿qué hay en la pantalla?

La tabla 0, que como ya hemos dicho ocupa 960 octetos, tiene

La memoria de video (I): texto y caracteres

Poseer 16K de VRAM es una característica a la que estamos poco o nada acostumbrados y manejarla bien es cuestión de práctica. Para ello y a lo largo de futuros números de la revista analizaremos cómo, donde y para qué almacena los datos en la pantalla la memoria de video en los cuatro modos SCREEN, al tiempo que mostraremos numerosas aplicaciones prácticas que se pueden emplear al programar.



como misión recoger los códigos de los caracteres que hay en la pantalla. Para poder manejarla, tendrás que poner a tu ordenador en el modo *SCREEN* 0, pues la memoria de vídeo obedece al sistema de almacenamiento que haya en ese momento en la pantalla, y si te hallas en el modo *SCREEN* 1 no podrás disponer de la tabla 0, sino de las tablas 5, 6, 7, 8 y 9, que son las que corresponden a ese modo de pantalla.

En el modo *SCREEN* 0, las 192 fi-

las × 256 columnas de píxeles se distribuyen de la siguiente manera: los píxeles se agrupan en cuadros de 8 filas × 6 columnas, dando lugar a un total de 24 filas × 40 columnas de cuadros, es decir, 960 cuadros (24 × 40 = 960). Cada uno de estos 960 cuadros se corresponde con cada uno de los 960 octetos existentes en la tabla 0, y en cada cuadro cabe una letra o un número.

Pero ya te habrás fijado en que al enchufar el ordenador no dispones de 40 columnas, sino de 37, y tampoco dispones de 24 filas, sino de 23, pues la última fila está destinada a mostrar las palabras que puedes emplear pulsando las teclas de función. Lo primero es debido a la orden *WIDTH*, utilizada para aumentar o disminuir el margen existente. Si, por ejemplo, tecleas:

WIDTH 10

los márgenes habrán aumentado tanto que sólo podrás escribir en un espacio de 10 columnas situado en la parte central de la pantalla. Precisamente eso es lo que indica el número situado tras la orden *WIDTH*, el número de columnas que han de quedar disponibles en la pantalla (en este caso 10). Eso es exactamente lo que ha sucedido a tu ordenador al enchufarlo: está bajo los efectos de un *WIDTH* 37 que ha sido ejecutado automáticamente al ser conectado. Pero como ya has visto, los márgenes pueden cambiar de tamaño. Si quieres disponer de toda la pantalla en *SCREEN* 0 sin márgenes, tendrás que teclear:

WIDTH 40

y observarás que del «Ok» que se ve en la esquina superior izquierda, no es visible la mitad de la «O», y tampoco se puede ver la mitad del cursor que está inmediatamente debajo. Esto sucede porque la mitad de la primera columna de cuadros no entren la pantalla. Por esto, y por darle una mayor estética a la presentanción, es por lo que los diseñadores de los ordenadores MSX decidieron establecer unos márgenes iniciales de *WIDTH* 37. Sin embargo, sin necesidad de utilizar la orden *WIDTH*, es posible escribir caracteres en los márgenes gracias a la tabla 0, como verás más adelante.

Los 960 octetos de la tabla 0 se dividen en 24 grupos de 40 octetos cada uno, correspondientes a cada una de las 24 líneas de la pantalla. Así, a cada uno de los 960 cuadros de la pantalla se le asigna un octeto. Si partimos de la base de que la tabla 0 comienza en el octeto número 1 será el del cuadro de la columna 1, línea 0: el octeto número 2 será del cuadro (2, 0); el octeto del cuadro (3, 0); y así podríamos seguir hasta el octeto número 39, correspondiente al cuadro (39,0) (véase la figura 1). Pero el octeto número 40 ya pertenece al grupo siguiente, al grupo de la línea 1, luego a éste octeto le corresponderá el cuadro (0,1); al octeto 41 el cuadro (1,1); al octeto 42 el cuadro (2,1); al octeto 43 el cuadro (3,1); y así hasta el octeto 79, correspondiente al cuadro (39,1). De esta manera el octeto 80 pertenece ya al grupo siguiente, al grupo de la línea 2, y siguiendo esta sucesión llegaríamos al octeto 920, primer octeto de la línea 23, y por tanto correspondiente al cuadro (0,23). Lógicamente, el último octeto de la tabla, el octeto 959, se corresponde con el cuadro de la última línea y la última columna: el cuadro (39,23).

La pantalla es el medio más importante que posée el ordenador para comuinicarse con el programador.

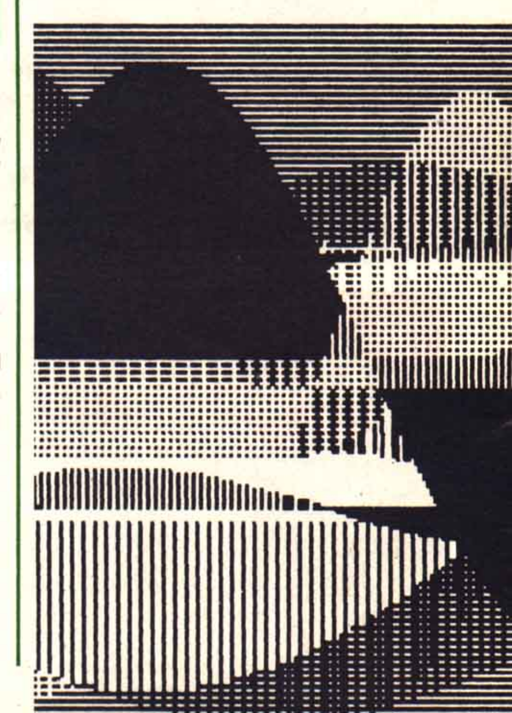
Si observamos detenidamente esta sucesión, podemos llegar a determinar una fórmula que logre obtener el octeto correspondiente a un cuadro en función de su línea, de su columna, y del primer octeto de la tabla 0:

primer octeto + n.° de línea × 40 + n.° de columna.

Todo esto está muy bien, pero tú te estarás preguntando hace rato para qué sirven los octetos y qué relación guardan con su cuadro correspondiente. Pues bien, los octetos tienen la misión de indicar qué carácter debe aparecer en su cuadro correspondiente. Así, si en un octeto se halla el número 65, en su cuadro correspondiente a la fuerza tiene que esta escrito el carácter 65, que es la «A» mayúscula. También es posible deducirlo al revés, es decir, si en un cuadro está escrito el carácter número 72, por ejemplo, (la «H» mayúscula) sin duda en su octeto se halla el número 72. (La lista con los caracteres y su número la puedes hallar seguramente en las instrucciones de tu ordenador).

La tabla 0 en la práctica

Vamos a comprobar que lo anteriormente expuesto es verdad. Continúa con tu ordenador en *SCREEN* 0 y establece uno márgenes de *WIDTH*



37. Si has borrado de la pantalla las instrucciones de las teclas de función, escríbelas de nuevo con el mando «KEY ON». Si no la has alterado, la primera de las intrucciones es «color» en minúsculas; si la has alterado, restitúyela con el mando «KEY 1, color». La primera letra, situada en el cuadro (2,23), es una «c» minúscula (carácter número 99). Para encontrar a qué octeto corresponde este cuadro, no hay más que aplicar la operación que antes determinábamos (primer octeto + n.° de línea × 40 + n.° de columna). En esta ocasión, 0 + 23 + 40 × 2 = 922, luego si tecleamos:

PRINT VPEEK (922)

o sea, «escribe el número contenido en el octeto 922», el resultado tendrá que ser 99, pues en su cuadro correspondiente se halla la «c» minúscula.

Si ahora queremos que en lugar de «color» ponga «dolor», sólo tenemos que establecer que en este cuadro hay una «d» minúscula (carácter número 100). Teclea, por lo tanto:

VPOKE 922,100

o sea, «introduce el número 100 en el octeto 922» y la palabra «dolor» estará en tu pantalla.

También gracias a la tabla 0 puedes escribir en los márgenes, sea cual sea su tamaño. Si ahora mantienes el *WIDTH* 37 (o establecer un *WIDTH* de menos columnas disponibles) y quieres escribir un carácter (pongamos por ejemplo la letra griega omega mayúscula, cárácter número 234) en la esquina superior derecha, que es el cuadro (39,0), tendrás que operar:

0 + 0 × 40 + 39 = 39

y a continuación situar el carácter en la pantalla:

VPOKE 39, 234

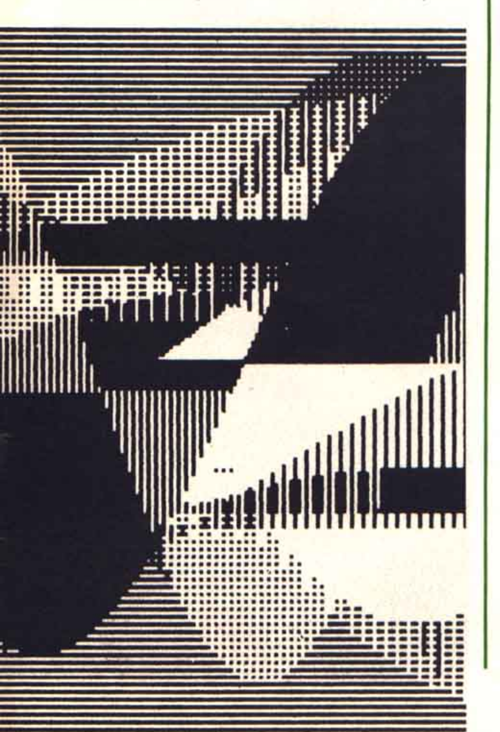
Y aún hay más utilidades en la tabla 0: si quieres llenar la pantalla de un carácter concreto, por ejemplo de ceros, puede ahorrar caracteres en las instrucciones en *BASIC*. Si no utilizaras la tabla 0, tendrías que teclear:

FOR A = 0 TO 959: VPOKE A, 48: NEXT

Con lo cual, aparte de tener que teclear menos, ahorras 46 octetos de la memoria RAM (no de la memoria vídeo) y además en el primer caso el ordenador tarda cerca de 6 segundos en visualizar la pantalla llena de ceros, y en el segundo caso sólo tarda 3 segundos.

Por último, sólo recordar que la tabla 0 es muy útil, pero no tiene por qué ser utilizada siempre que se vaya a escribir algo en la pantalla. Habrá casos de textos cortos en los que no hará falta usar las tabla 0, bastará con la orden *PRINT*.

Gracias a la facilidad de acceso a la memoria de vídeo, hacerse un juego de caracteres es sencillo.





La tabla 2: ¿qué debe aparecer en la pantalla?

La tabla 2, segunda y última fila del sistema de almacenamiento de datos de *SCREEN* 0, fue ya tratada en una de sus aplicaciones: los Caracteres Programables, en el número 3 de MSX Magazine. Su longitud es de 2048 octetos, y está destinada a guardar los patrones de los 256 caracteres que pueden aparecer en la pantalla. En nuestros ordenadores MSX la tabla 2 comienza en el octeto 2048, por lo que termina en el 4095, ambos inclusive. Para averiguar dónde comienza en tu ordenador, utiliza la instrucción «BASE (2)», como antes hemos explicado; y si a ese primer octeto de la tabla le sumas 2047, tendrás el último octeto de la tabla.

Pero esta tabla, a diferencia de la anterior, tiene una dificultad en el sistema de almacenamiento. Los patrones de los caracteres que retine en la memoria están previstos para cuadros de 8 filas × 8 columnas de píxeles, no para cuadros como los del modo *SCREEN* 0, con 8 filas × 6 columnas de píxeles. Hay, por lo tanto, 2 columnas de cada patrón que no aparecen en pantalla, y que son las dos columnas de la derecha.

Una vez considerado esto, pasemos a explicar cómo funciona la tabla 2. Sus 2048 octetos se dividen en 256 grupos de ocho octetos, que se corresponden con cada uno de los 256 caracteres que pueden apa-

recer en la pantalla. Si los números introducidos en estos 8 octetos se transforman al sistema de numeración binaria, se convierten en las 8 filas de píxeles de cada cuadro, correspondiéndose cada cifra en binario con cada píxel.

La operación con la que podemos llegar a deteminar el primer octeto del grupo de ocho de un carácter es:

primer octeto de la tabla + n.° de carácter × 8

La tabla 2 en la práctica

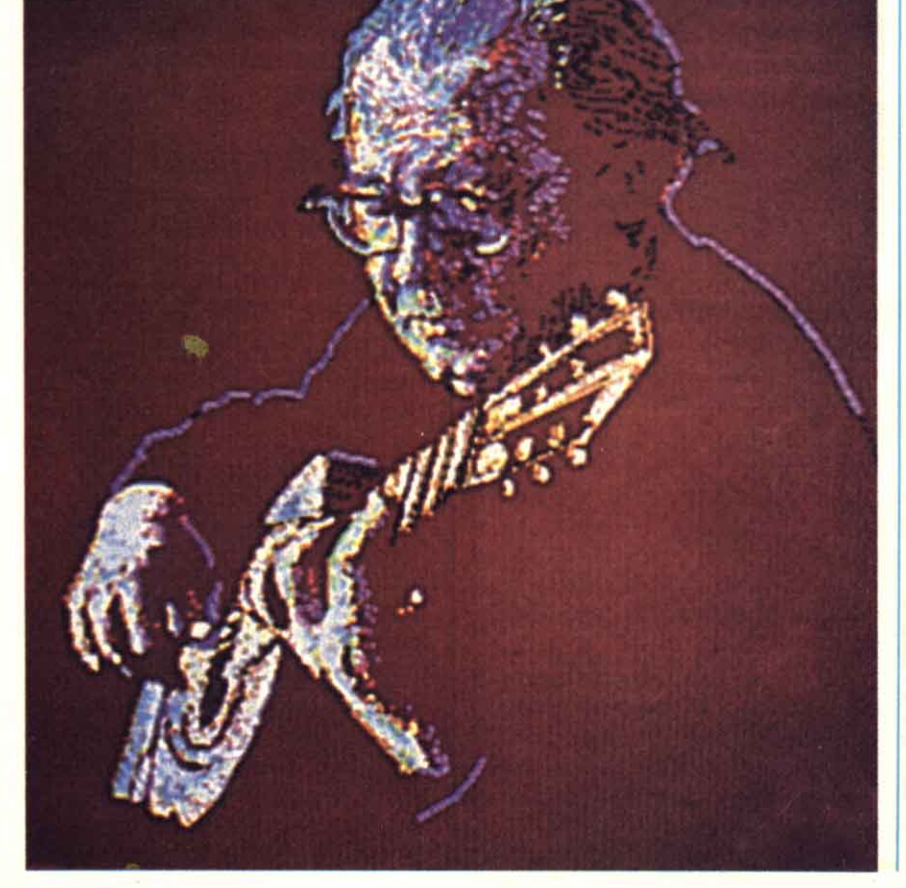
Aparte de los caracteres programables, la tabla 2 tiene muchas más aplicaciones, como es el cambio de forma de los caracteres de la pantalla. Esta aplicación está basada en que el más mínimo cambio que sufran los los patrones repercute instantáneamente y con todas sus consecuencias en la pantalla. Este programa es un ejemplo de lo dicho:

```
10 SCREEN 0
20 LOCATE 3,10: PRINT «aaaaaaaa»
30 VPOKE 2825,&B01110000: VPOKE 2826,&B00001000: FOR T=0 TO 100: NEXT
40 VPOKE 2825,0: VPOKE 2826,&B01110000: FOR T=0 TO 100: NEXT: GO TO 30
```

Las «a» de la línea 20 (que han de ser FORZOSAMENTE MINUSCULAS) son las que, si ejecutas el programa, parecen moverse. Esto es debido a unos cambios efectuados en el patrón de la «a» minúscula, que comienza, como puedes comprobar, en el octeto 2824 (es el carácter 96) y termina en el octeto 2831.

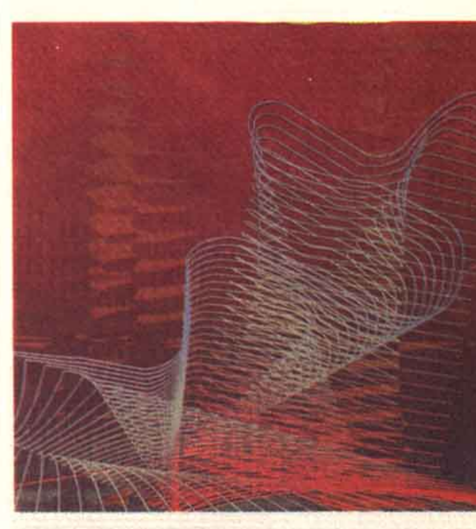
En los dos primeros *VPOKE* de la línea 30, el trazo superior de la «a» minúscula, que normalmente está en el octeto 2826, se eleva una línea de píxeles, situándose en el octeto 2825. A continuación se incluye un bucle *FOR-NEXT* ralentizador, sin el cual este movimiento sería demasiado rápido y confuso. Después hay otras dos órdenes *VPOKE* en la línea 40, que dejan al patrón de la «a» minúscula tal y como aparece siempre, con el trazo superior en el octeto 2826. Al ir alternando armónicamente estos dos patrones de la «a» minúscula, dá la impresión de que la «a» se encuentra realizando un movimiento.

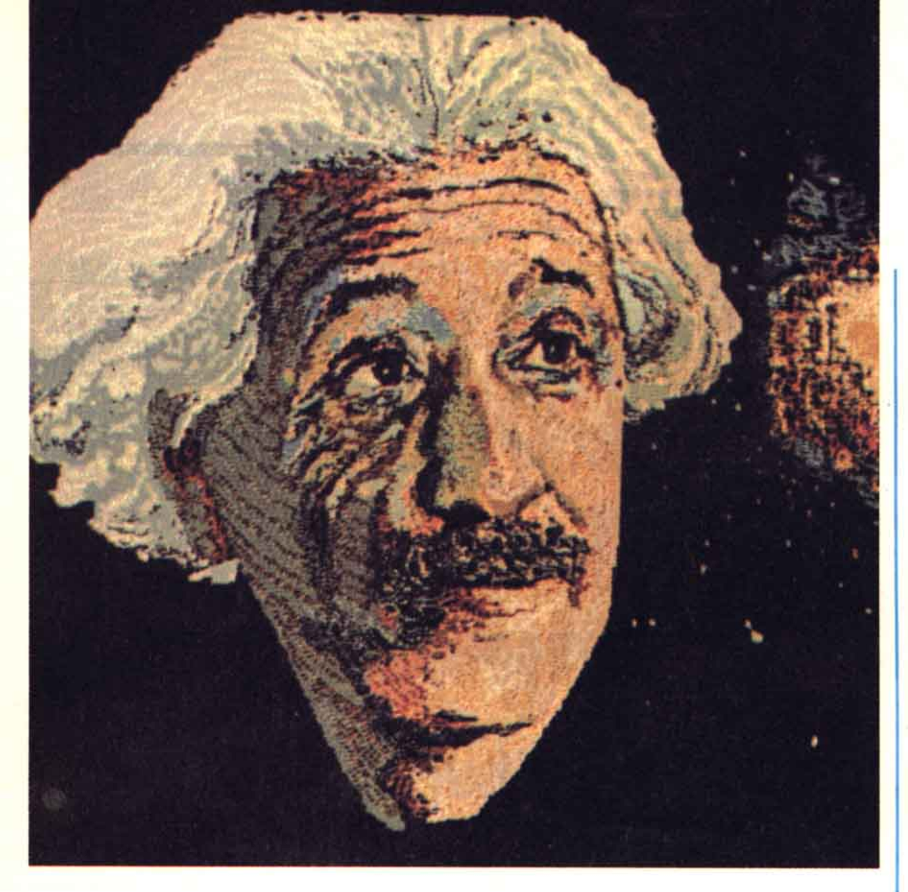
Gracias a la alternativa de patrones parecidos, es posible lograr una sensación de movimiento muy real, como ya has podido ver. Ahora te retamos a que intentes hacer tú tus propios movimientos, no sin antes darte unas cuantas ideas, como por ejemplo girar las letras en uno u otro sentido, ensancharlas y estrecharlas, alargarlas o acortarlas, hacerlas explotar, o todo lo contrario, hacerlas «implotar», es decir conseguir que la letra se retraiga sobre sí misma, quedándose del tamaño de un píxel, y a continuación borrar ese píxel, etc...

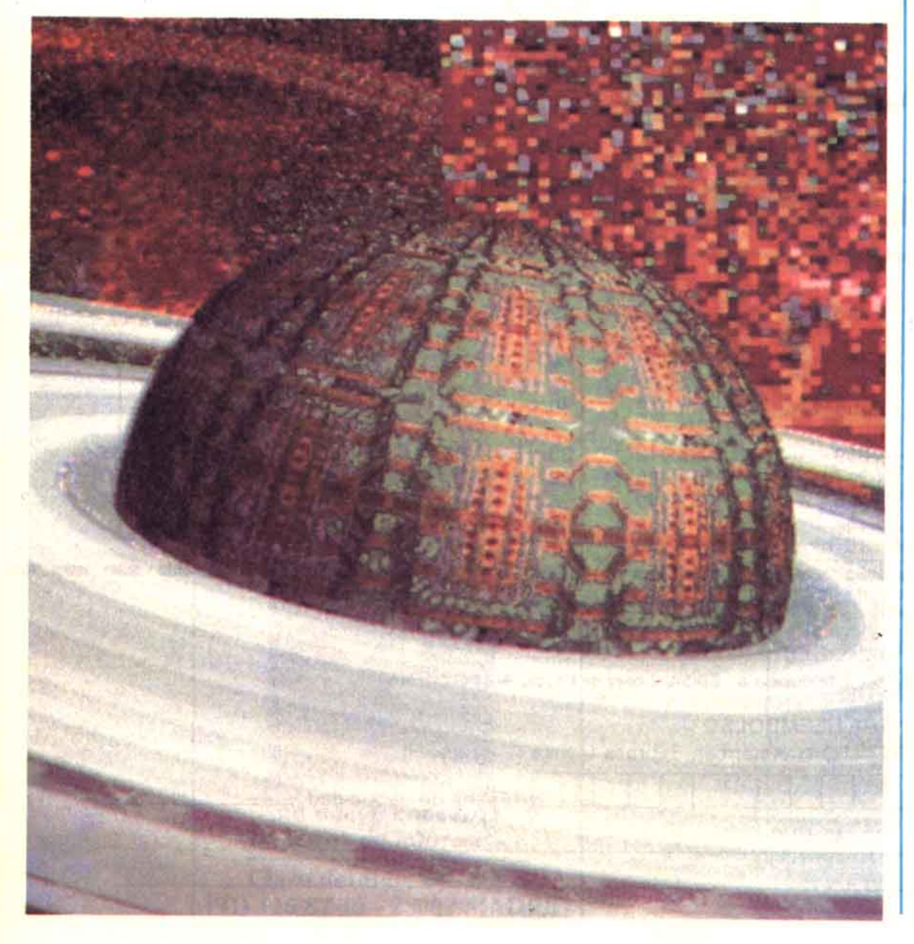


Combinando las dos tablas

Dejando de lado las múltiples aplicaciones de las dos tablas por sepa-







rado, también existen muchos efectos que se pueden lograr combinando las dos tablas adecuadamente. Uno de estos efectos es el efecto *FLASH*, que ya existe en otro microordenadores, y al que se pueden acceder en los ordenadores MSX gracias a la memoria de vídeo. El efecto *FLASH* consiste en un parpadeo armónico de los colores de fondo y de tinta de una parte de la pantalla, o en toda ella. Esto se puede lograr en *SCREEN* 0 creando los caracteres negativos de los que vayan a ocupar la zona de la pantalla en la que queramos crear el efecto *FLASH* y situando armónicamente los caracteres negativos y los positivos en los mismos cuadros de la pantalla.

Para crear los caracteres negativos de otros lo único que hay que hacer es restar a 255 la fila de píxeles correspondiente en forma de número binario de ocho cifras. Por ejemplo, si tenemos el número 120 que es binario es &B01111000, y queremos hallar su equivalente en negativo, sólo tenemos que operar:

255-129 = 135

y tendremos su equivalente en negativo en sistema decimal, porque si ahora 135 lo transformamos a binario, será &B10000111, y lo que eran ceros son ahora unos, y lo que eran unos son ahora ceros.

Este sistema de transformación a negativos es el utilizado en el siguiente programa:

```
10 SCREEN 0: WIDTH 37: KEY ON
20 FOR R=1 TO 3: READ Q: P=2048 + Q × 8
30 FOR T=P TO P + 7: VPOKE T, VAL («&B» + BIN$ (255 - VPEEK (T - 256)): NEXT: NEXT
40 DATA 109,115,120
50 VPOKE 299,109: VPOKE 300,115: VPOKE 301,120
60 FOR T=1 TO 200: NEXT
70 VPOKE 299,77: VPOKE 300,83: VPOKE 301,88
80 FOR T=1 TO 200: NEXT: GOTO 50
```

Este programa emula el efecto *FLASH* en la siglas «MSX» que aparecen en la zona central de la pantalla. De la línea 20 a la 40 genera, gracias

a la tabla 2, tres nuevos caracteres («M», «S», y «X» mayúsculas en negativo), por el método antes explicado, que sustituyen a los patrones de las letras «m», «s» y «x» minúsculas. En la línea 50 sitúa los nuevos caracteres en un lugar central de la pantalla, gracias a la tabla 0, y la línea 70 sitúa en ese mismo lugar las mayúsculas en positivo. Las líneas 60 y 80 son bucles *FOR-NEXT* retardadores.

Hasta aquí hemos examinado cómo almacenan los datos las tablas de *SCREEN* 0. en el próximo número enalizaremos las tablas de *SCREEN* 1 y cómo, gracias a ellas, se puede lograr el efecto *FLASH* con dos cualesquiera de los 16 colores.

José María Cavanillas

	COLUMNA 0	C 1	C 2		C 37	C 38	C 39
LINEA 0	OCTETO 0	1	2		37	38	39
L 1	40	41	42		77	78	79
L 22	880	881	882		917	918	919
L 23	920	921	922		957	958	959



(12 NUMEROS)

TARIFA DE PRECIOS DE SUSCRIPCION

	CORREO ORDINARIO		CORREO CERTIFICADO		CORREO AEREO		CORREO AEREO-CERTIF.	
	PTAS.	$	PTAS.	$	PTAS.	$	PTAS.	$
ESPAÑA	3.000	21	3.276	23	3.048	22	3.324	24
EUROPA, MARRUECOS, TUNEZ, TURQUIA, ARGELIA Y CHIPRE	3.456	25	4.272	30	3.564	25	4.380	31
COSTA RICA, CUBA, CHILE, PARAGUAY Y R. DOMINICANA	3.396	24	4.212	30	3.972	28	4.788	34
GIBRALTAR Y PORTUGAL	3.264	23	4.080	29	3.115	22	3.931	28
FILIPINAS	3.264	23	3.540	25	3.583	25	3.859	27
RESTO DEL MUNDO	3.456	25	4.272	30	4.032	29	4.848	34

CUPON DE PEDIDO

Recorte y envíe este cupón a: **EDISA. Lopez de Hoyos, 141 - 28002 - MADRID**

El importe lo abonaré: POR CHEQUE □ CONTRA REEMBOLSO □
CON TARJETA DE CREDITO □ American Express □ Visa □ Interbank □

Número de mi Tarjeta: □□□□□□□□□□□□□□□□□□ Fecha de caducidad: ________

NOMBRE ________________________________

DIRECCION ________________________________

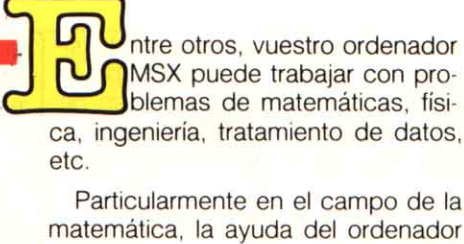
CIUDAD ________________ D.P. ____ PROVINCIA ________________

La matemática y el ordenador

Como podreís imaginar los ordenadores se inventaron para algo muy distinto que para jugar. Cierto es que los ordenadores personales actualmente en el mercado han sido preparados con miras a poder realizar con ellos todo tipo de juegos, pero no obstante su fundamento es el mismo que el de los grandes ordendores destinados al tratamiento de enormes masas de información. De hecho, vuestro ordenador MSX puede manejar información como lo hacen los grandes ordenadores profesionales y además pueden ser programados para realizar juegos gráficos y sonoros.

La gran diferecia entre uno y otro tipo de ordenador está en la cantidad de información que pueden manejar simultáneamente. Un ordenador MSX podrá tratar menos información que un gran ordenador profesional, pero puede tratarla de la misma forma.





Entre otros, vuestro ordenador MSX puede trabajar con problemas de matemáticas, física, ingeniería, tratamiento de datos, etc.

Particularmente en el campo de la matemática, la ayuda del ordenador puede ser muy valiosa. El tratamiento de un problema matemático mediante un ordenador se realiza con el auxilio de los que llamamos ANALISIS NUMERICO. El Análisis Numérico consiste en el estudio de un problema matemático que puede o no ser resuelto de forma téorica. Lo que hace el Análisis Numérico es ANALIZAR un tipo de problema determina- [illegible] RITMO o secuencia de operaciones con el que se pueden tratar de forma general problemas concretos.

La colaboración del ordendor en estos temas, consiste en realizar el trabajo penoso y repetitivo, ahorrando así mucho tiempo al investigador (o al estudiante).

El ordenador puede resolver en segundos o minutos problemas que resueltos de forma manual pueden llevar días, meses o años.

Sin embargo, el ordenador no es un ser inteligente, por lo que necesita que en cada momento se le indique cómo tiene que tratar los datos que se le han suministrado. El ordenador no es, por tanto, un fin, sino un medio.

Pero... veamos un ejemplo de Análisis Numérico y cómo el ordenador debe manejar la información:

Cálculo de los ceros de una función real de una variable real (1)

Entre las funciones reales de una variable real, es decir, aquellas de la forma:

$$y - f(x) \qquad x \varepsilon R, \; y \varepsilon R \; (1)$$

hemos aprendido a hallar los valores de x (variable independiente) que hacen a y (variable independiente) igual a cero, es decir los valores de x que satisfacen la ecuación:

$$f(x) = 0 \qquad (2)$$

dentro de un intervalo en que f(x) es continua y derivable.

Sin embargo, podemos observar que sólo podemos resolver un número determinado de estas ecuaciones. Por ejemplo, sabemos resolver las siguientes ecuaciones:

$$3x - 2 = 0 \qquad (3.1)$$
$$x^2 - x - 1 = 0 \qquad (3.2)$$
$$\varepsilon^x - 1 = 0 \qquad (3.3)$$
$$\text{sen } z = 0 \qquad (3.4)$$
$$x^3 + 2x^2 - x - 2 = 0 \qquad (3.5)$$

Algunas de estas ecuaciones tienen más de un cero. (Llamamos ceros de una función y – f(x) a los valores de x qu e satisfacen dicha función para y = 0). Por ejemplo, los valores $x = 0$, $x = \pi$, $x = 2\pi$... satisfacen la ecuación (3.4) y los valores $x = 1$, $x = -1$, $x = -2$ satisfacen la ecuación (3.5).

Sin embargo no conocemos métodos exactos para calcular los ceros de las siguientes funciones:

$$y = \varepsilon^x - \text{sen } x \qquad (4.1)$$
$$y = \log x^2 - 3x \text{ arc tg } x \qquad (4.2)$$
$$y = x^6 - 3x^5 + x^4 - 0{,}5x^3 - 2x^2 + x - 1 \qquad (4.3)$$

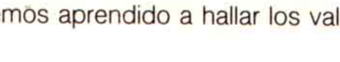
En realidad, el número de ecuaciones que no pueden resolverse por métodos exactos es enorme.

Para resolver este tipo de problemas se han ideado diversos métodos iterativos: Regla de Newton, Regla de Fourier, Regula Falsi...

Estos métodos iterativos son métodos de aproximación que utilizan un valor (x_1) de x para obtener otro (x_2) que esté más cerca de la solución (x_0) que el anterior. El valor de x así hallado (x_2) se utiliza a su vez para encontrar otro (x_3) más próximo a x_0 que x_2 (y que x_1), y así sucesivamente.

Por regla general, mediante estos métodos llegaremos a obtener un va-





lor x^n muy próximo a x_0. Normalmente tomaremos este resultado (x = xn) como exacto, al no poder se calculado (o no interesar) la solución exacta ($x = x_0$). Veamos esto con un ejemplo:

Supongamos que queremos hallar un valor de x que satisface a la ecuación

sen x = 0 (5)

que esté en el intervalo (2.4).

Sabemos por trigonometría que x = π y generalmente lo expresaremos así, pero no podremos expresar nunca el valor exacto de x.

Sin embargo podremos considerar como válidas cualquiera de las siguientes aproximaciones:

$x_1 = 3$
$x_2 = 3{,}1$
$x_3 = 3{,}14$
$x_4 = 3{,}142$
$x_5 = 3{,}1416$

.

.

.

$x_{10} = 3{,}141592654$

El hecho de que nos conformemos con $x_1 = 3$ ó prefiramos tomar el valor $x_{10} = 3{,}141592654$ como exacto depende sólo del grado de exactitud con que queramos trabajar. Generalmente las calculadoras de bolsillo dan un máximo de 12 cifras, por lo que no tendrá sentido, en caso de que trabajemos con ellas, tomar el valor más exacto de

$$x = 3{,}1415926535898$$

con el que trabajan los ordendores.

Por tanto, en la mayoría de los casos no precisaremos encontrar el valor exacto x_0 de la solución a la ecuación:

$$f(x) = 0, \qquad x \in (a,b)$$

y nos contentaremos con un valor suficientemente próximo para nuestros propósitos.

Supongamos que sabemos que la función y = f(x) tiene un cero entre las abscisas x = a y x = b (b a), y que este cero es $x = x_0$.

Trataremos de encontrar un valor de x = xn suficientemente próximo a x_0. Para ello impondremos a x una determinada condición, por ejemplo que el valor absoluto de $f(x_n)$ sea menor que una determinada cantidad suficientemente pequeña, es decir:

$$|f(x_n)| < \varepsilon \qquad (6)$$

O bien que la diferencia entre x_n y el valor calculado anteriormente, x_{n-1} sea, en valor absoluto, menor que ε, es decir:

$$|x_n - x_{n-1}| < \varepsilon \qquad (7)$$

Esto equivale a decir que hasta una cierta cifra x_n y x_{n-1}, son iguales. Por ejemplo, en el caso que hemos visto anteriormente, si tomamos $\varepsilon = 10^{-3}$ (ε = 0,001) tomaríamos como válidad la solución:

$x_4 = 3{,}142$
ya que siendo
x 3,1416, es:
$|x_5 - x_4| = |3{,}14159 - 3{,}1416| = 0{,}0004 < 0{,}001$.

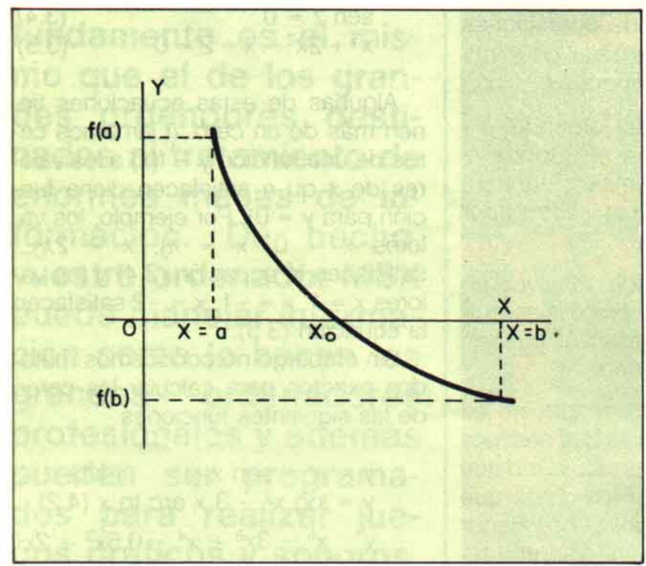


Figura 1

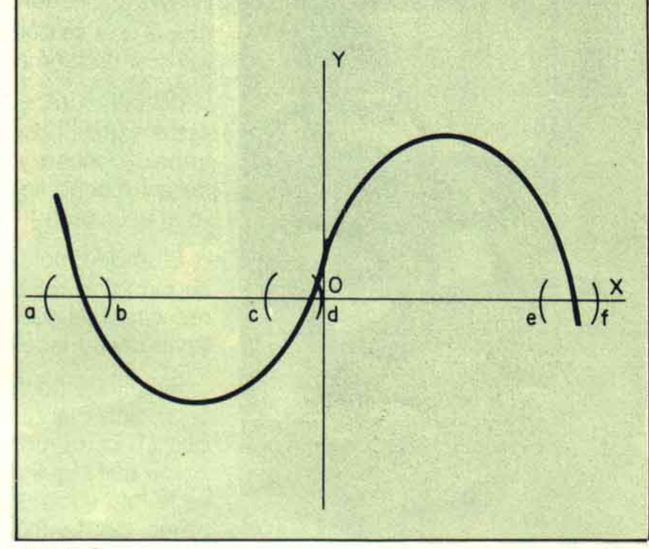


Figura 2

Podemos imponer otras condiciones a x_n, en función de nuestras necesidades. Nosotros tomaremos aqui la condición (6).

Tras esto, calcularemos la solución x_n siguiendo este esquema:

1. Tomamos un valor inicial para x, siguiendo $x_1 = a$.
2. Calculamos $f(x_1)$
3. Si $|f(x_1)| < \varepsilon$, x_1 es la solución buscada, y aquí termina el problema.
4. Si $|f(x_1)| > \varepsilon$, tomaremos un nuevo valor de x, x_2 calculado a partir de x_1 según el método con el que se esté trabajando. Con este nuevo va-

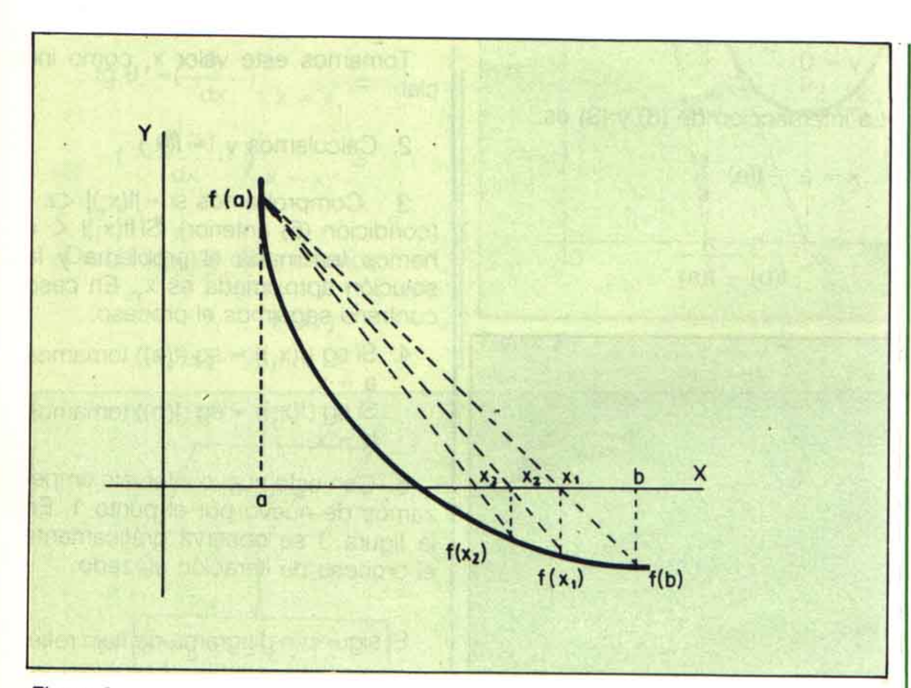


Figura 3

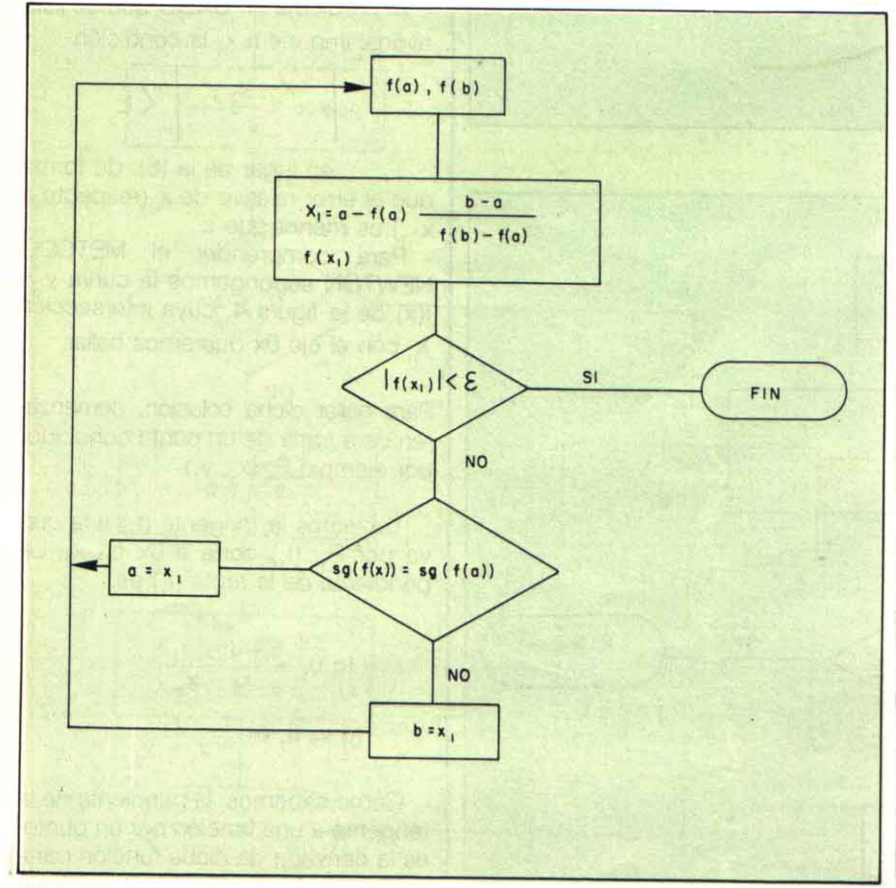


Diagrama 1

lor de x volveremos a empezar por el punto 2 hasta que se cumpla la condición impuesta.

Observamos que estos métodos exigen conocer un intervalo (a,b), donde se encuentre al menos una solución a la ecuación (2). (En realidad esto no es exactamente cierto. Si en (a,b) no existe ninguna solución, el método calculará otra solución próxima a los valores extremos del intervalo).

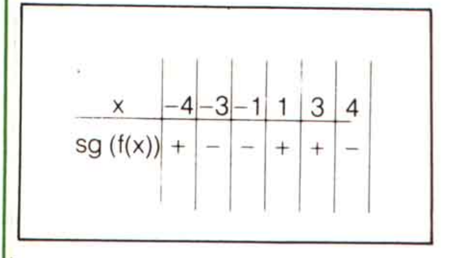
Sabemos por el Teorema de Bolzano, que si en el intervalo (a,b) existe un cero de la función $y = f(x)$ (o un número impar de ceros), el signo de f(a) es contrario al de f(b), es decir:

$$sg\,(f(a)) \neq sg\,(f(b))$$

(Ver figura 1)

Este teorema nos permite «acotar» los ceros de una función, es decir, hallar intervalos (a,b), (c,d)... en los que haya un cero de la función. (Figura 2).

Esto lo haremos tomando valores de x_1 y calculando f (x). Por ejemplo para $y = \text{sen}\ x$, podemos elaborar la siguiente tabla:



x	−4	−3	−1	1	3	4
sg (f(x))	+	−	−	+	+	−

y así sabremos que en cada uno de los intervalos:

$I_1 = (-4, -3)$
$I_2 = (-1, 1)$
$I_3 = (3, 4)$

hay al menos un cero de $y = f(x)$.

La REGULA FALSI resuelve el problema de encontrar una solución (x) a la ecuación $f(x) = 0$ en el intervalo (a,b) de la siguiente forma: (fig. 3)

1. Tratamos la recta que une los puntos (a, f(a)) y (b,f(b)) (secante) y

calculamos su intersección con el eje 0x.

La ecuación de dicha recta es:

$$\frac{x - a}{y - f(a)} = \frac{b - a}{f(b) - f(a)}$$

Y la del eje 0x:

$$y = 0$$

La intersección de (8) y (9) es:

$$x = a - f(a) \frac{b - a}{f(b) - f(a)}$$

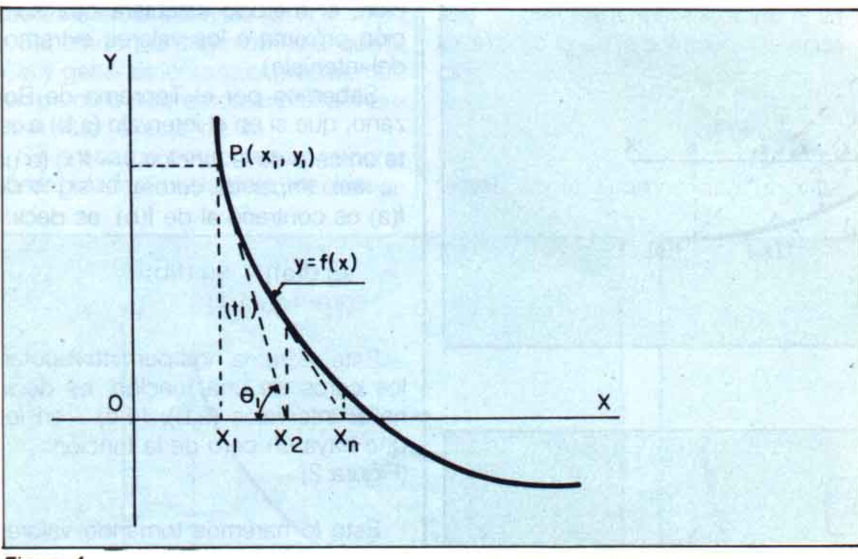


Figura 4

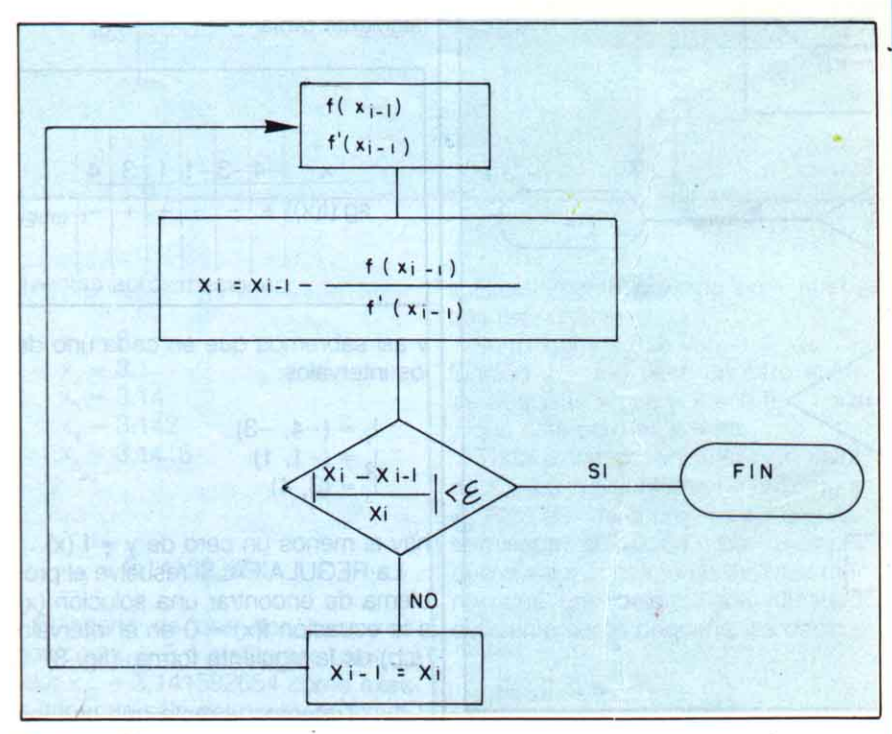


Diagrama 2

Tomamos este valor x_1 como inicial.

2. Calculamos $y_1 = f(x_1)$

3. Comprobamos si $= |f(x_1)| < \varepsilon$ (condición (6) anterior). Si $|f(x_1)| < \varepsilon$ hemos terminado el problema y la solución aproximada es x_1. En caso contrario seguimos el proceso.

4. Si sg $(f(x_1))$ = sg $(f(a))$ tomamos $a = x_1$
   Si sg $(f(x_1))$ = sg $(f(b))$ tomamos $b = x_1$

5. Con este nuevo intervalo empezamos de nuevo por el punto 1. En la figura 3 se observa gráficamente el proceso de iteración utilizado:

El siguiente diagrama de flujo refleja esquemáticamente el proceso llevado a cabo por el método:

El problema en BASIC que se lista al final impone a x_1 la condición

$$\left| \frac{x_i - x_{i-1}}{x_i} \right| < \varepsilon$$

en lugar de la (6), de forma que el error relativo de x_i (respecto a x_{i-1}) es menor que ε.

Para comprender el METODO NEWTON supongamos la curva y = f(x) de la figura 4, cuya intersección x_n con el eje 0x queremos hallar.

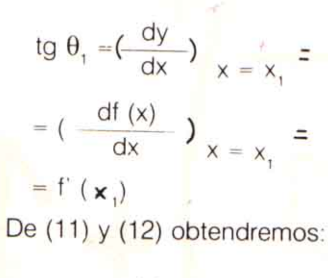
Para hallar dicha solución, comenzaremos a partir de un punto conocido, por ejemplo P_1 (x_1, y_1)

Trazamos la tangente (t_1) a la curva por P_1. (t_1) corta a 0x en x_2. La pendiente de la recta (t_1) es:

$$\text{tg } \theta_1 = \frac{-y_1}{x_2 - x_1}$$

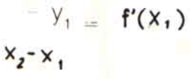
$$(\text{ó tg } \theta_1 = \frac{y_1}{x_1 - x_2})$$

Como sabemos, la pendiente de la tangente a una función por un punto, es la derivada de dicha función particularizada por dicho punto, es decir, en nuestro caso:



$$\operatorname{tg} \theta_1 = \left(\frac{dy}{dx}\right)_{x = x_1} = \left(\frac{df(x)}{dx}\right)_{x = x_1} = f'(x_1)$$

De (11) y (12) obtendremos:



$$\frac{-y_1}{x_2 - x_1} = f'(x_1)$$

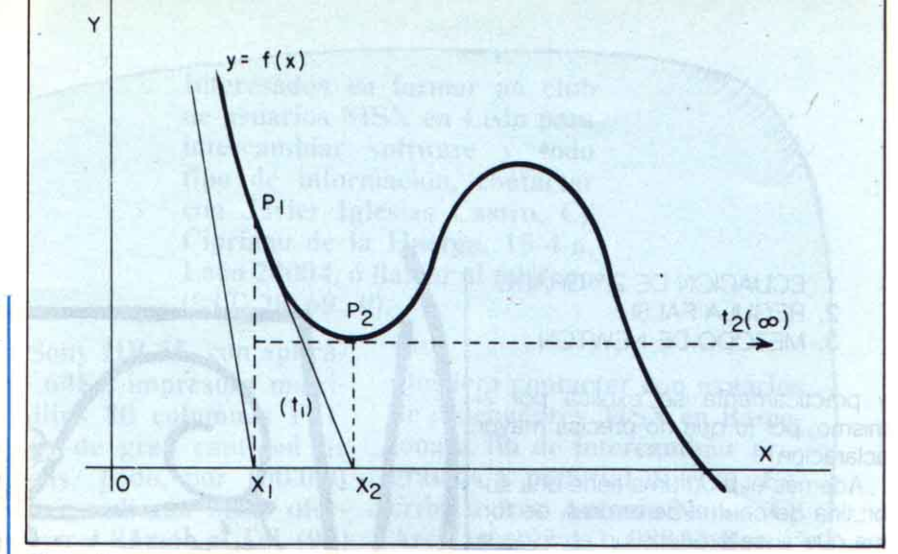


Figura 5

a,b,c

b = - b / 2 a
c = - c / a

d = b² - a.c

d < Ø

SI

NO

d = √d

d = √-d

X1 = -b + d
X2 = -b - d

X1 = - b + i d
X2 = - b - i d

FIN

Diagrama 3

y como $y_1 = f(x_1)$, $f(x_1) = f'(x_1)(x_1 - x_2)$

o bien:

$$x_1 - x_2 = \frac{f(x_1)}{f'(x_1)}$$

Por tanto:

$$x_2 - x_1 = \frac{f(x_1)}{f'(x_1)}$$

Esta es la fórmula a la que llegó Isaac Newton.

De forma más general podemos calcular el valor x_i de la iteración i a partir del de la iteración i−1:

$$x_i = x_{i-1} - \frac{f(x_{i-1})}{f'(x_{i-1})}$$

Para finalizar el proceso de aproximación impondremos a x_i la condición:

$$\left|\frac{x_i - x_{i-1}}{x_i}\right| < \varepsilon$$

El siguiente diagrama de flujo explica de forma simplificada el proceso seguido por el programa en BASIC que se edita la final:

Como habreis observado este tratamiento es más sencillo que el de la Regula Falsi, pero requiere conocer no solo f(x) sino también sus derivadas, cosa que no es demasiado complicada. Sin embargo el método fallará cuando f' (x) = 0 en el punto considerado, es decir cuando la tangente sea paralela la eje 0x (ver fig. 5).

El programa en BASIC que se lista a continuación, contiene el siguiente MENU:

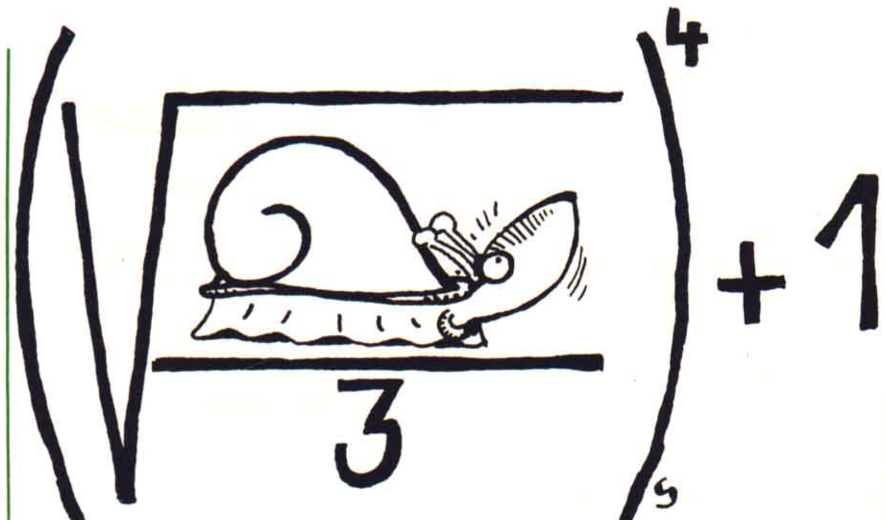
1. ECUACION DE 2.° GRADO
2. REGULA FALSI
3. METODO DE NEWTON

y prácticamente se explica por sí mismo, por lo que no precisa mayor aclaración.

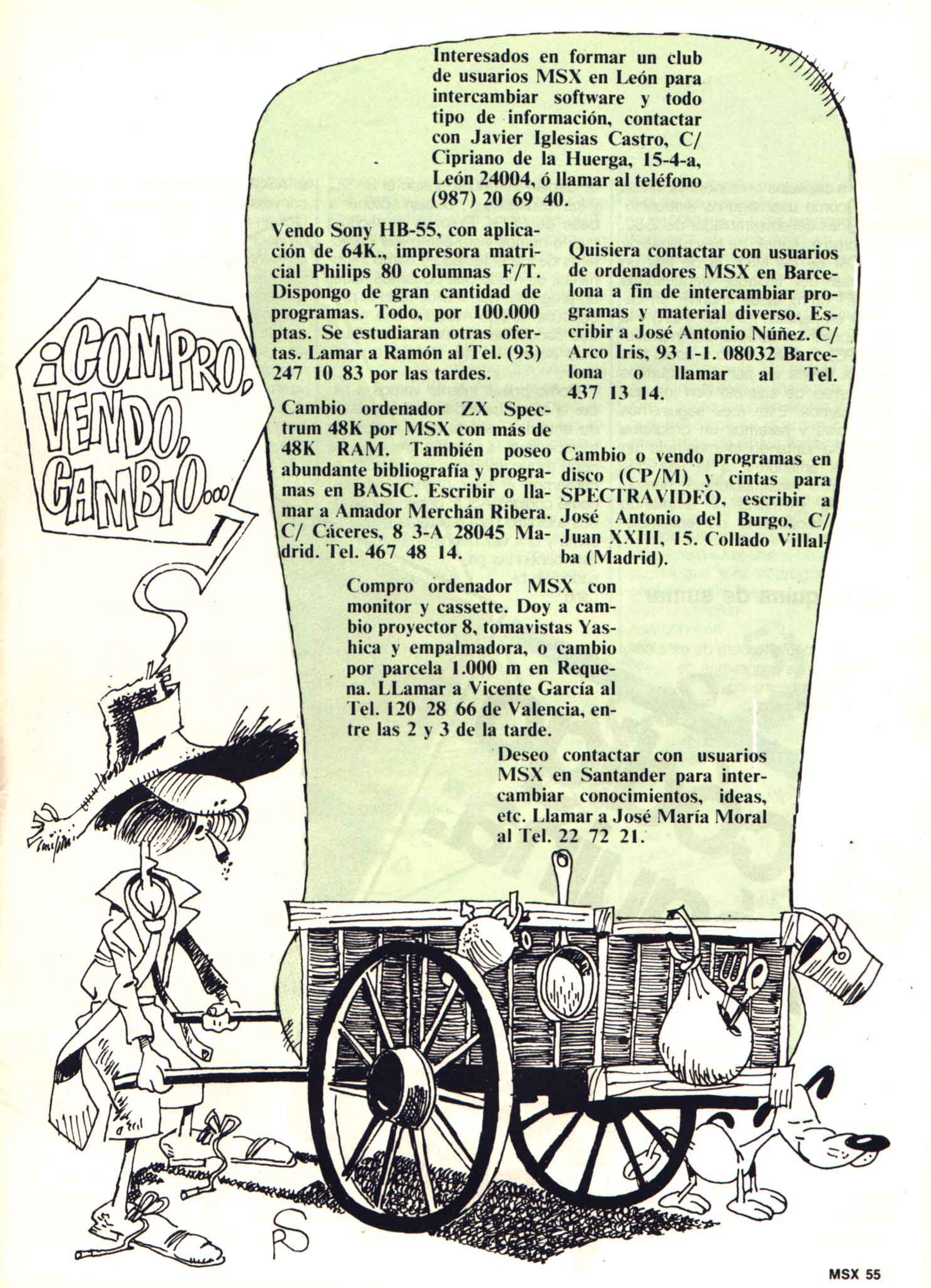
Además el programa tiene una subrutina de control de errores, de forma que si se produce un error de la forma «división por cero», el programa no se interrumpe y dá como solución aproximada el valor de x calculadc anteriormente a la producción de dicho error.

La opción 1 calcula los valores de la función $y = ax^2 + bx \times c$

Es decir, calcula sus dos racices, ya sean éstas reales o imaginarias según el siguiente diagrama de flujo:



Referencias:
— J. Rey Pastor y A. Castro: Elementos de Matemáticas.
— A. Apóstol: Calculus.
— J.C. Mason: Basic: Numerical Mathematics.



Interesados en formar un club de usuarios MSX en León para intercambiar software y todo tipo de información, contactar con Javier Iglesias Castro, C/ Cipriano de la Huerga, 15-4-a, León 24004, ó llamar al teléfono (987) 20 69 40.

Vendo Sony HB-55, con aplicación de 64K., impresora matricial Philips 80 columnas F/T. Dispongo de gran cantidad de programas. Todo, por 100.000 ptas. Se estudiaran otras ofertas. Lamar a Ramón al Tel. (93) 247 10 83 por las tardes.

Quisiera contactar con usuarios de ordenadores MSX en Barcelona a fin de intercambiar programas y material diverso. Escribir a José Antonio Núñez. C/ Arco Iris, 93 1-1. 08032 Barcelona o llamar al Tel. 437 13 14.

Cambio ordenador ZX Spectrum 48K por MSX con más de 48K RAM. También poseo abundante bibliografía y programas en BASIC. Escribir o llamar a Amador Merchán Ribera. C/ Cáceres, 8 3-A 28045 Madrid. Tel. 467 48 14.

Cambio o vendo programas en disco (CP/M) y cintas para SPECTRAVIDEO, escribir a José Antonio del Burgo, C/ Juan XXIII, 15. Collado Villalba (Madrid).

Compro ordenador MSX con monitor y cassette. Doy a cambio proyector 8, tomavistas Yashica y empalmadora, o cambio por parcela 1.000 m en Requena. LLamar a Vicente García al Tel. 120 28 66 de Valencia, entre las 2 y 3 de la tarde.

Deseo contactar con usuarios MSX en Santander para intercambiar conocimientos, ideas, etc. Llamar a José María Moral al Tel. 22 72 21.

El código máquina: ese desconocido

En capítulos anteriores ya vimos como usar algunas instrucciones del ensamblador del Z-80, así como el manejo de tres rutinas de la *ROM* de los MSX, la de leer un carácter del teclado, denominada *CHGET* (9Fh); la de imprimir un carácter en pantalla *(CHPUT*, A2K) y la de imprimir un carácter por impresora *(LPOUT*, A5h). Junto a estas tres rutinas dimos un par de pequeños programas de ejemplo con los que practicamos. Este mes seguiremos usándolas y haremos un programa de suma juntando lo mostrado en articulos anteriores, a la vez que mostramos instrucciones nuevas que nos permitirán ir ampliando nuestro vocabulario de ensamblador.

La máquina de sumar

En el capítulo tercero de esta serie hicimos varios programas de suma. El problema común a todos ellos era que los datos se debían introducir desde el *BASIC* y los resultados se debían obtener a base de *PEEK*. Pero en el capítulo cuarto hemos visto como leer caracteres desde el teclado y como imprimirlos, de modo que podamos hacer un programa que realice todo el proceso de código máquina si a nuestros conocimientos previos le añadimos algún otro que vamos a explicar.

Como primer intento vamos a hacer la suma de modo que los datos de entrada y salida sean dados en hexadecimal. La razón de esto se debe a que los datos leidos que, recordemos el mes pasado, se leen como códigos ASCII; deben ser pasados a binario para realizar la suma y una vez hecha ésta deben ser pasados de binario a ASCII para imprimir. La primera conversión de ASCII a binario, resulta fácil en ambos sistemas (incluso más en decimal)pero en cambio el proceso inverso que pasa de binario a ASCII el resultado, resulta más fácil si lo queremos representar en hexadecimal, necesitando incluso realizar divisiones u otras operaciones complejas, por lo que lo dejaremos para más adelante. De momento veamos como realizar el proceso tal como hemos dicho.

(Cap. 5)



Como primer paso supondremos que los sumandos y el resultado son de un *byte*. Lo que indica que se le van a meter números comprendidos entre Oh y FFh (255). Una vez que realicemos este programa, lo ampliaremos para más *bytes* (es decir, números más grandes).

En primer lugar se debe realizar el proceso de entrada de los datos, que se podría hacer con la rutina *CHGET* ya vista y almacenando los caracteres en alguna zona de memoria, pero en su lugar utilizaremos una nueva llamada a otra rutina del sistema operativo, que hace este proceso de coger los caracteres y los almacena en memoria automáticamente. El formato de estos datos de entrada es del tipo «xx+yy» donde xx es el primer número a sumar (que debe tener como máximo dos cifras del «0» al «F») e yy es el segundo, que debe cumplir los mismos requisitos que el primero. Para simplificar más aún, el programa supondrá que no se van a meter espacios entre estos números y el signo «+» ni al principio ni al final y por último los dígitos de «A» a «F» deberán ir en mayúsculas. Posteriormente veremos como hacer evitar que esta reglas sean tan estrictas. La rutina que hemos comentado es *QINLIN*, que después de imprimir una interrogación en pantalla, aceptauna línea desde el teclado, terminando cuando se pulsa la tecla *RETURN* o *STOP*. Esta línea es almacenada en un *buffer* cuya dirección inicial viene dada en el par de registros *HL*. Antes de seguir adelante explicaremos que es un *buffer*, ya que son ampliamente utilizados en ensamblador y resultando muy útiles en multitud de situaciones.

Recordará que los programas que hacemos se almacenan en memoria de forma consecutiva a partir de una dirección inicial. Este mismo sistema puede ser utilizado para guardar datos de longitud indefinida. Empezando a almacenarlos a partir de una dirección y llenando la memoria según vengan más. Un detalle especial de estos *buffer* (o memorias temporales, como también se les llama) es que los datos se guardan como los programas de abajo hacia arriba, de manera que si el primero se alamcena en la 40500, el segundo lo hará en la 40501 (en decimal naturalmente), el tercero en la 40501, etc. Debiendo existir un sistema que nos indique cual es el último dato introducido, ya que si no el programa seguirá hacia abajo leyendo memorias con contenidos aleatorios, creyendo que eran datos. En la rutina *QINLIN* este fin de indica almacenado un cero (el valor binario, no el código ASCII 48, que es el correspondiente al símbolo «0») después del último carácter tecleado antes de pulsar *RETURN*. La dirección de inicio del *buffer* se indica, como dijimos antes, por medio de la dirección almacenada en *HL* cuando la rutina nos devuelve el

Y TU ¿NO LE TIENES MIEDO AL CODIGO MAQUINA?
¿MIEDO YO? CON ESTOS ARTICULOS DE LA MSX MAGAZINE...
¡¿MIEDO YO?!..)
¡¡NUNCA!!

control. Esta dirección (que la elije el ordenador en función de sus posibilidades de memoria y que es independiente de la que use nuestro programa) es una inferior a la del primer carácter almacenada. De modo que si este está en la 40500 como dijimos antes, *HL*, contendrá 40499 y el final de los datos lo marca el cero en la parte superior. Para llamar a esta rutina basta hacer un *CALL OB4h* (180), que es su dirección de llamada, con lo cual se leen los caracteres del teclado y se almacena en memoria, devolviendonos en *HL* la dirección que comentábamos antes.

Una vez realizada esta entrada tendremos en memoria una lista de caracteres ASCII correspondientes a las teclas pulsadas. Hemos de recordar que en esta lista no estarán directamente los números a usar, ya que si el primero que hemos introducido ha sido «D4», en memoria no tendremos un byte con este valor, sino dos *bytes* en los cuales estarán introducidos los códigos ASCII de los dos dígitos. Es decir, 68 y 52 por este orden. El problema que se nos plantea es, pues, pasar estos dos *bytes* (o lo que sean) a un valor binario. Para ello pensemos que D4h quiere decir Dh*Fh+4h (13*16+4), ó dicho de otro modo D0h+4h (208+4), el proceso que podremos usar será obtener el valor binario de cada uno de los dígitos (13 y 4) y a continuación coger el primero, si hay otra cifra detrás, multiplicarlos por dieciséis (decimal) y al resultado sumarle el segundo, con lo que obtendremos el valor buscado. Si en cambio sólo hay un dígito antes del símbolo «+», querrá decir que sólo hay unidades y por tanto no hay que multiplicar ni sumar nada más. Con el segundo sumando el proceso es igual con la excepción de que el fin del número no lo marca el símbolo «+» (de código ASCII 43) sino el código 0, introducido por *QINLIN* al final.

Una vez descompuesto el proceso en oeraciones más sencillas, veamos como realizar éstas. En primer lugar, pasemos a códigos ASCII a valores binarios. Para ello basta fijarse que los códigos ASCII de los dígitos del 0

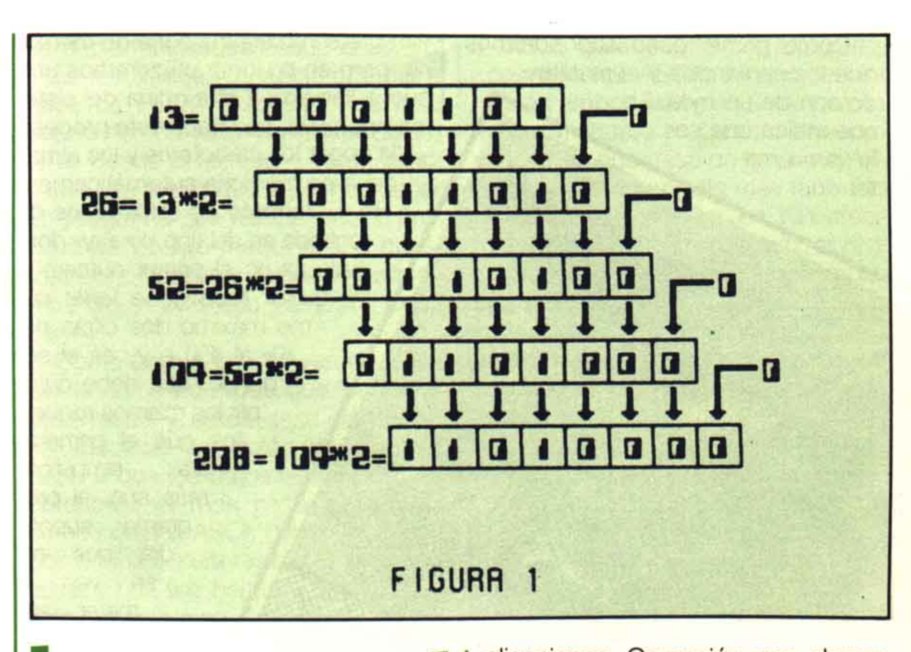


FIGURA 1

Cualquier tipo de programa realizado se almacena de forma consecutiva a partir de una dirección inicial.

al 9 son consecutivos empezando con el cero en el 48 y acabando con el 57 del nueve. Con los valores ASCII de la «A» a «F» también sucede lo mismo pero empezando por 65, para la «A», por tanto si el código correspondiente es inferior a 65, querrá decir que es un dígito del 0 al 9 con lo que para pasarlos al binario les restamos 48, obteniendo su valor. Es decir, 48-48 dá 0, 49-48 dá 1, 50-48 dá 2, 51-48 es tres y 52-48 es 4, como queremos. Si en cambio el dígito es uno de los caracteres de «A» a «F», habrá que restarle 55, ya que esto dará 10 para la «A», 11 para la «B», etecetera.

Para obtener el valor final hay que realizar un proceso algo más complicado, ya que hay que realizar multiplicaciones. Operación que, desgraciadamente, el Z-80 no sabe realiazar directamente pero que como veremos es muy sencilla de implementar, sobre todo si hay que multiplicar por una potencia de 2, como es este caso en el que hay que multiplicar por Fh (16). Para ello se utiliza una propiedad de los números que consiste en que multiplicar por la base es lo mismo que introducir un cero a la derecha. Como ejemplo podemos ver que en decimal, multiplicar por diez es añadir un cero a la derecha y 123 por 100 es 1230. En binario el proceso es el mismo pero con la diferencia de que la base es dos y por tanto se multiplica por este valor. De modo que si queremos multiplicar 9 (1001b) por 2 (0010b), basta añadirle un cero a la derecha al nueve operando en binario, que dá 18 (10010b). Sabiendo esto, podemos hacer unas cuantas operaciones artiméticas para ver como se multiplica un número «X» por Fh (16). Sabemos que X*16 es lo mismo que (X*8)*2 y esto es ((X*4)*2)*2, que es equivalente a (((X*2)2)*2)*2. Por lo que la operación queda reducida a multiplicar el número cuatro veces sucesivas por dos como se ve en la figura 1.

Para realiza esta conversión en lenguaje máquina, primero seleccionamos una memoria (que llamamos

*DATO*1 y situaremos en la dirección 40500) en la que se guardará el primer número e introducimos en ella un cero como valor inicial. A continuación cogemos el primer carácter que se haya almacenado en el *buffer* y se mira si está entre cero y nueve, si es así se le resta 48, y si no 65. Al resultado de este proceso se le suma el valor de la memoria *DATO*1 que es inicialmente cero, por lo que no varía el resultado obtenido. A continuación se coge el segundo dígito y si no es el símbolo de suma «+», se realiza con él la misma operación que con el primero (ahora almacenado en *DATO*1) pero previamente se multiplica este por 16 de modo que cuando llegue el momento de sumar el contenido de dicha memoria al resultado de esta segunda conversión, ya tendremos hecha la multiplicación correspondiente para que el resultado sea correcto. Un detalle a tener en cuenta de este proceso de conversión es que no se comprueba si los dígitos leídos están realmente comprendido entre el «0» y el «9» o entre la «A» y la «F», de modo que si introducimos cualquier otro carácter (que tampoco sea «+») el programa no lo comprobará y actuará como si fuese realmente un dígito numérico, generando resultados imprevistos.

Para verlo más claramente realizaremos el proceso figuradamente su-



poniendo que llamamos a *QINLIN* y nos devuelve nuestra entrada, en la que el primer valor es el *D4h* que pusimos de ejemplo y que vamos a convertir explicando paso a paso las intrucciones que hay que realizar. En primer lugar cargamos la dirección de *DATO* 1 en l registro *IX*, ya que todos los accesos a esta memoria y a las posteriores a ella, en las que se almacenan los números a sumar y el resultado, los haremos indexando, ya que nos dá más versatilidad (hay más funciones que funcionen con este tipo de direccionamiento que si se indica directamente la dirección) y permite cambiar la dirección donde se almacenan los datos con sólo cambiar la instrucción que almacena la dirección en *IX*. Esta dirección es *LID, dato* 1. A continuación se mete un cero en la memoria *DATO*1 y en la siguiente a ella (la 40501, en la que se carga un cero en el acumulador por medio de la instrucción *LD A, hh* (que como vimos, almacenaba en el acumulador el número colocado inmediatamente detrás de la instrucción) y se guarda en la memoria *DATO*1 por medio de la instrucción *LD (IX+0)*, en la que el registro *IX* más cero nos da 40500. Para gurdar otro cero en la siguiente, se usa la misma instrucción, pero en este caso el desplazamiento (el número que se suma a *IX* es 1) nos queda *LD (IX+0),A.*

Una vez hecho esto, se carga en el acumulador el código correspondiente al primer carácter leído. Este carácter está en la posición superior a la indicada por *HL*, por lo que primero debemos incrementar éste por medio de la instrucción *INC HL* (que ya usamos en otro capítulo con el registro *IX*) y se leé por medio de la instrucción *LD A, (HL)*, que también hemos visto. Una vez hecho esto incrementaremos *HL* (23h ó 35) en uno para que se quede apuntando a la memoria que contenga el siguiente carácter. Esto se hace muy sencillamente usando otra vez la instrucción *INC HL*. Una cosa a tener en cuenta con todas las instrucciones que realizan incrementos o decrementos es

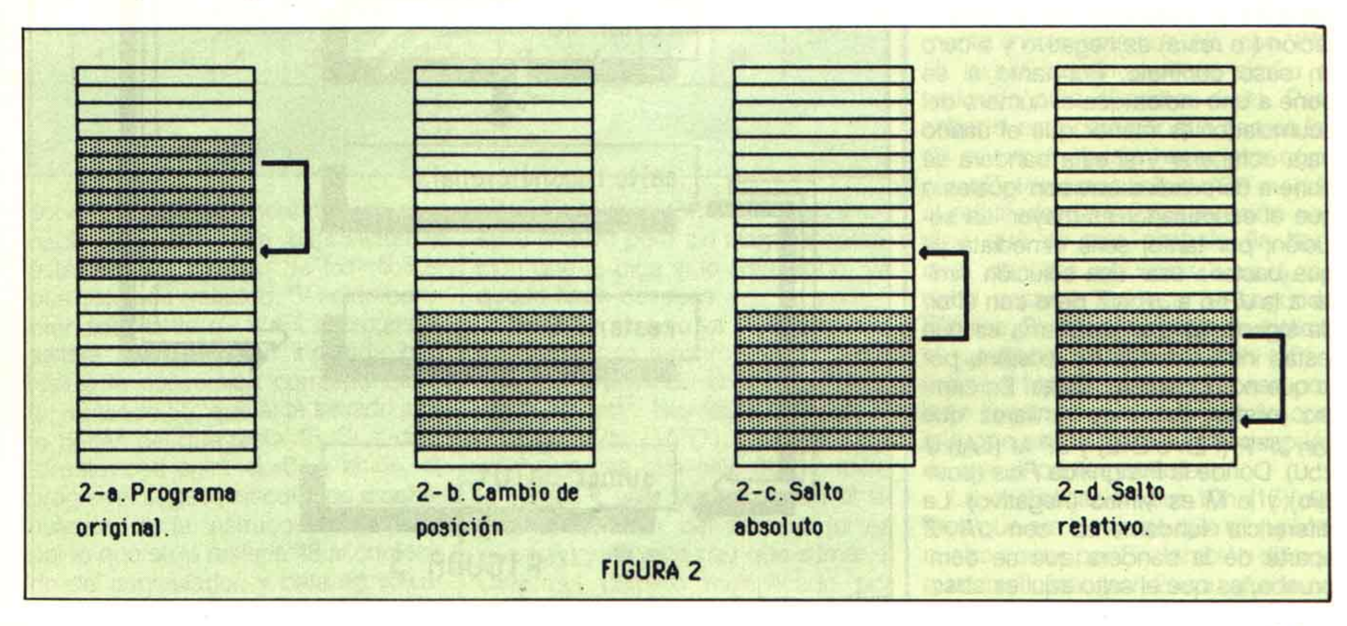


FIGURA 2

que éstos son cíclicos, es decir, si estamos trabajando con 8 *bits* y tenemos el número *FFh*, al incrementarle daría 100h, pero al no caber en el registro quitaría el dígito superior quedando 00h y a partir de ahí volvería a subir. Lo mismo sucede con los registros de los 16 *bits* que pasan de FFFFh a 0000h y viceversa. Esto no importa en nuestro caso, pero en otros programas puede ser importante.

Con el carácter correspondiente al primer dígito en el acumulador realizamos una comparación con 65. Como dijimos, una comparación es básicamente una resta del contenido del acumulador menos el dato (65 en este caso) pero sin alterar el contenido del acumulador y modificando solamente las banderas del Z-80. si el resultado de esta comparación es negativo, indica que es un dígito del 0 al 9, si es positivo (se considera el cero como positivo ya que tiene el *bit* más significativo a cero como los números positivos) estamos con un dígito de «A» a «F». A continuación de esta instrucción hay que realizar una bifurcación según el tipo, para restarle el valor correspondiente. Para ello utilizaremos el salto condicional que ya vimos. Pero en lugar de hacerlo con la bandera Z que nos indicaba si los dos números son iguales, usaremos la bandera S (de signo) que se pone a 1 si el resultado de la comparación (o resta) es negativo y a cero en caso contrario. Por tanto si se pone a uno indica que el número del acumulador es menor que el usado para comparar y si esta bandera se pone a cero indica que son iguales o que el acumulador es mayor. La solución, por tanto, sería inmediata ya que bastaría usar una solución similar a la *JR* o a *JR NZ* pero con el *bit* de signos. El gran problema es que ¡estas instrucciones no existen!, por lo que no podemos usarlas. En cambio existen dos muy similares que son *JP P* (F2h ó 242) y *JP M* (FAh 0 250). Donde la P significa *Plus* (positivo) y la M es *Minus* (negativo). La diferencia fundamental con *JR Z* aparte de la bandera que se comprueba, es que el salto aquí es absoluto y no relativo. Es decir, si con *JR* (salto relativo) el dato que se le daba era una dirección de 8 *bits* relativa a la posición en ese momento del contador del programa. Con el *JP* (salto absoluto) lo que se le da es un número de dieciseís *bits* que indica la posición a saltar dentro de la memoria del ordenador, independientemente de la posición del contador de programa en ese momento. Pero aparte de esta diferencia, que explicaremos ahora, el funcionamiento es igual y las podemos usar sin ningún problema. En este caso usaremos *JP P.*'

Esta diferencia entre saltos absolutos y realtivos se puede ver en la figura 2, en la que nos hemos supuesto que existen dos programas distintos en memoria. En la figura 2-a se representa un programa que está en una zona de la memoria y realiza un salto a otra dirección dentro del mismo programa, situada cuatro *bytes* (por ejemplo) más abajo. Si este programa le cambiamos a otra zona de la memoria como se ve en 2-b, pueden suceder dos cosas. Si el salto era relativo sucede que, como se

Los programas que utilizan direcciones absolutas no se pueden cambiar de la zona de memoria para la que está previsto que funcionen.

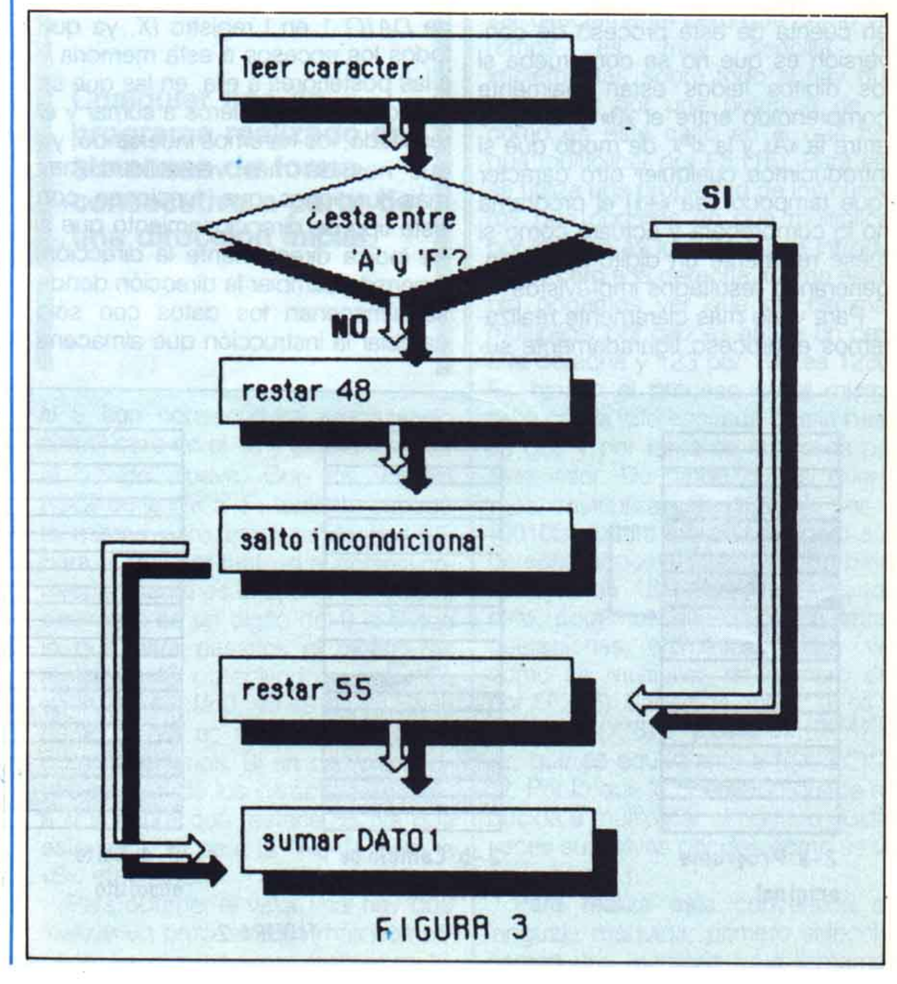


FIGURA 3

muestra en la figura 2-c, el salto se realiza a otra dirección distinta que al principio, pero que también se encuentra cuatro *bytes* por debajo de donde se salta y por tanto está la misma instrucción de antes. En cambio si el salto es absoluto, como se muestra en la figura 2-d, se sigue saltando a la misma dirección inicial, pero al haber cambiado el programa de posición no se encontrará la instrucción que esperábamos y se producirá un error que puede generar múltiples consecuencias; que se cuelgue el ordenador, que el programa de resultados erróneos o cualquier otra cosa. Por tanto, los programas que utilizan direcciones absolutas (como el que estamos desarrollando ahora) no se puden cambiar de la zona de memoria para la que está previsto que funcionen.

Antes de esta pequeña disertación sobre los saltos comentábamos que había de hacer una bifurcación y ésta será en función de los dígitos que estemos tratando. Si están comprendidos entre «A» y «F», el programa saltará a la dirección indicada que realiza la conversión correspondiente, restando 55, que está situado justo detrás del que resta 48. Si el dígito estuviese entre el «0» y el «9», el programa seguirá ejecutando a continuación de la instrucción de salto, por lo que se le restaría 48 al contenido del acumulador, y para no volver



a restar el 55 correspondiente a los caracteres «A» a «F», se salta al proceso a que viene después (para ver esta sucesión de saltos al mirar la figura 3) por medio de una instrucción *JR* (18h o 24), que realiza un salto relativo pero sin ninguna condición que le diga si lo hace o no, ya que lo hace después.

Después de esto ya tenemos en el acumulador (o registro «A» del Z-80) el dígito expresado en binario. Después de esto hay que sumarle el contenido de *DATO*1. La razón de esta suma es que este mismo ciclo lo vamos a usar para procesar el siguiente carácter del número (si es que hay) y en este hay que sumar el valor del primero multiplicado por dieciséis, valor que como veremos vamos a almacenar en *DATO*1. Esta suma se realiza con una operación que ya hemos visto, *ADD A,* (IX+0), que suma al contenido del acumulador el de la direción indicada por *IX* más cero.

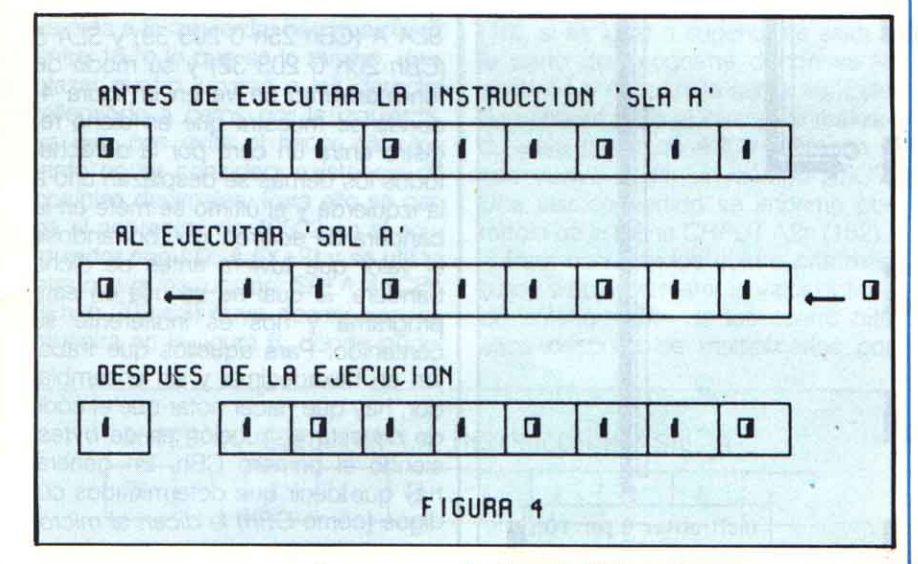


FIGURA 4

Ahora hay que mirar si el siguiente carácter tecleado es un «+» u otro dígito del número, por lo que tendremos que leer, éste en el registro A, pero antes debemos salvar su contenido (que es el resultado de la conversión) para no destruirlo al leer el nuevo *byte*. Podríamos guardalo en la memoria que tenemos reservada (*DATO*1) pero como luego es posible que tengamos que multiplicarlos por 16, tendríamos que volver a leerlo. Para evitar movimientos inútiles lo que haremos será guardarlo en otro registro del microprocesador. El «B». Este además nos permite que la multiplicación se haga en él, lo que nos evitará problemas. La instrucción para pasar el dato del registro «A» al «B» es *LD B,A* (47h o 71), similar en funcionamiento a las otras instrucciones *LD* que vimos pero que se diferencia en que en este caso no hay que poner ningún *byte* de datos después de la instrucción, ya que el origen y el destino van especificados en la misma instrucción.

A continuación se carga el siguiente carácter en «A» por medio de la instrucción *LD A, (HL)* y seguidamente incrementamos *HL* con *INC HL* siguiendo un proceso similar al realizado para el primer dígito. Para saber si este carácter que hemos leído es un «+» basta hacer otra comparación de este símbolo (CP «+»). Si el resultado de la comparación nos dice que sí este símbolo (la bandera de cero «Z», está a uno), se debe pasar a procesar el segundo número. Si no es, hay que multiplicar el resultado anterior por 10h (16) y volver a repetir todo el proceso con este segundo carácter, para lo cual hay que saltar hacia arriba después de leer el primer carácter.

La multiplicación por 10h (16) dijimos que era lo mismo que multiplicar cuatro veces por 2 y también dijimos que multiplicar por 2 era añadir un

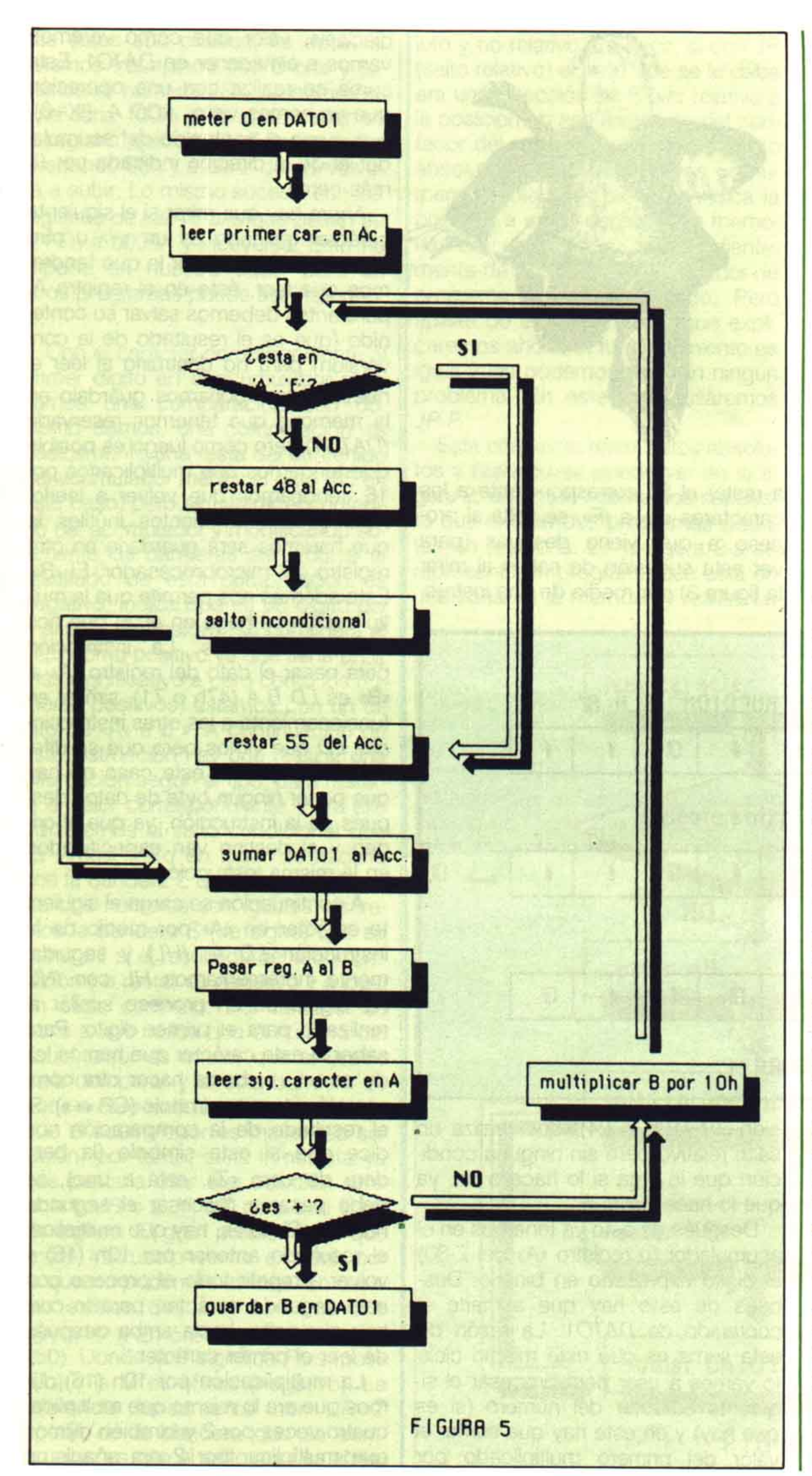


FIGURA 5

En las operaciones lógicas, el resultado no depende del resto de los bits, en las aritméticas, si.

cero a la derecha (corriéndose todas las demás cifras a la izquierda una posición y perdiéndose la situada más a la izquierda). Pues bien, hay una instrucción que hace exactamente eso y además nos permite hacerlo con el acumulador o con el registro «B». Estas instrucciones son *SLA A* (CBh 25h 0 203 39) y *SLA B (CBh* 20h 0 203 32) y su modo de funcinamiento se ve en la figura 4, donde se muestra que en dicho registro entra un cero por la derecha, todos los demás se desplazan uno a la izquierda y el último se mete en la bandera de acarreo «C», borrándose el valor que tuviera antes de dicha bandera, la cual no se usa en este programa y nos es indiferente su contenido. Para aquellos que trabajen en hexadecimal y sin ensamblador, hay que hacer notar que el código de esta instrucción es de *bytes*, siendo el primero CBh. En general hay que decir que determinados códigos (cómo *CBh) le dicen al microprocesador que esa instrucción es de 16 bits*, al contrario que la mayoría que son de 8, por decirlo de algún modo son prefijos telefónicos que nos permiten acceder a otra parte del conjunto de instrucciones. Por tanto *CBh* no significa nada por si sólo, si va seguido de 27h significa *SLA A* y si nos encontramos el 27h suelto sin dicho prefijo significará *DAA* que es otra instrucción que veremos en capítulos posteriores.

El organigrama completo de todo el proceso de conversión de un número de *ASCII* a binario se da en la

figura 5, donde podemos ver todos los procesos y saltos comentados antes.

La conversión para el segundo dato se realiza exactamente igual que en caso del primero pero utilizando la memoria situada a continuación de *DATO*1 (40501) y la comparación que indica el final del número ahora será el código 0h (0) que es el que coloca *QINLIN* al final.

Una vez convertidos los dos números, se suman sencillamente con la instrucción *ADD* que ya vimos en el capítulo 2 y almacenaremos el resultado en 40502 por medio de la instrucción *LD* (IX+2),A.

La última parte que nos queda es la conversión de binario a *ASCII* y su impresión a pantalla. Para ello se ha de realizar el proceso inverso al realizado anteriormente. En primer lugar y para imprimir el código correspondiente al dígito más significativo (que va más a la izquierda) hay que dividir entre 16, o lo que es lo mismo, desplazar cuatro lugares a las derecha rellenando a ceros por la izquierda, ya que nos evita el hacer cálculos tanto no se considera existencia de posibles decimales. Para ello se carga el contenido de 40502 en el acumulador con *LD, A* (IX+2) y se utiliza otra nueva instrucción: *SRLA A* (CBh 3Fh ó 203 63) cuya descripción se muestra en la figura 6, donde podemos ver que es justo al contrario de *SLA A*, ya que por la izquierda del *byte* entran ceros y los *bits* que salen por la derecha van a la bandera de acarreo. Una vez hecho esto volvemos a realizar el mismo proceso de comparación, esta vez con *Ah* (10), si es igual o superior se salta a la parte del programa donde se le suma 55 y si no se le suma 48. Este proceso es justo el inverso al realizado para pasar de *ASCII* a binario y nos vuelve a dar un código *ASCII*. Una vez convertido se imprime por medio de la rutina *CHPUT* A2h (162).



Para convertir los cuatro *bits* más bajos se podría restar el valor obtenido anteriormente de los cuatro *bits* altos después de multiplicarlos por 10h, pero hay un sistema mucho más sencillo como vamos a ver a continuación.



Todos aquellos que hayan estudiado algo de lógica sabrán lo que es una función *AND*. Para aquellos que no lo sepan diremos que es una operación lógica en vez de aritmética. Llamamos operaciones artiméticas a aquellas que operan sobre todo el *byte* acarreando resultados de un *bit* al siguiente, por ejemplo, la suma de una operación artimética ya que cuando sumamos dos *bits* que valen 1, el resultado es 0 y acarreamos un 1 a la siguiente posición. En cambio las operaciones lógicas no acarrean resultados de una posición a otra, sino se operan individualmente sobre cada *bit* (o pareja de *bits* si son dos operandos) y el resultado de cada *bit* no depende de los demás. Una vez definido esto diremos que un *AND* de dos *bits* es una operación que da de resultado 1, sólo si los dos *bits* valen 1. En cualquier otro caso el resultado es 0. Cuando hacemos el *AND* de dos *bytes* (como es el caso

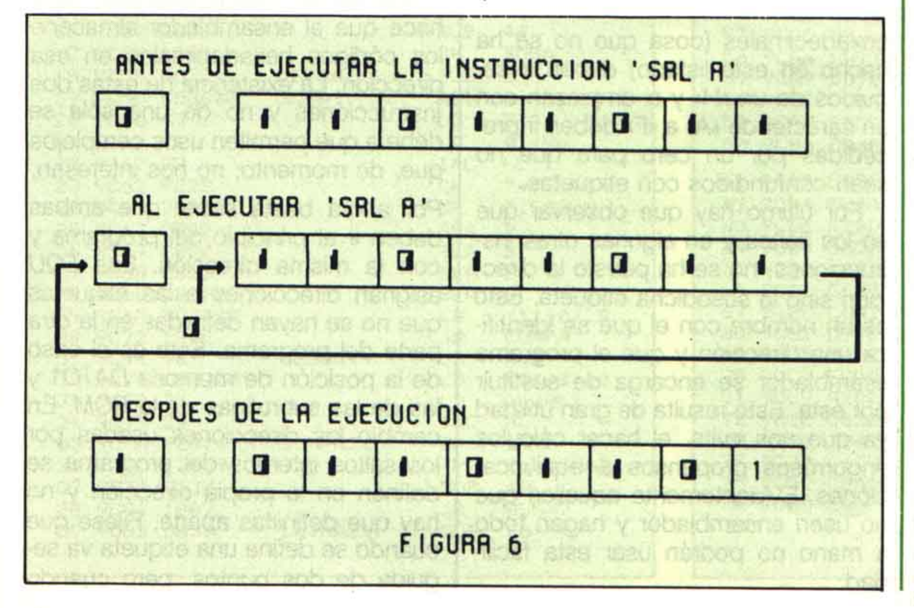


FIGURA 6

de la instrucción que vamos a ver), lo que estamos haciendo es un *AND* de cada una de las parejas de *bits*. Los dos primeros, los dos segundos, etc. Esta instrucción es muy útil entre otras cosas para poner determinados *bits* a cero de un *byte*, ya que si hacemos un *AND* de un *byte* (como el número binario con el que estamos trabajando) con otro preparado especialmente (por ejemplo, el 00001111) llamado «máscara», aquellos *bits* de nuestro dato que estuviesen a cero se siguen quedando a cero, pero en cambio los que estuviesen a uno se quedarían a uno sólo si el *bit* correspondiente de la máscara estuviese a uno. Con ello y utilizando la máscara que acabamos de indicar (como se ve en la figura 7, en la que utilizamos de dato D4h), podemos poner a cero los cuatro *bits* superiores obteniendo el número menos significativo que se convierte a *ASCII* y se imprime igual que antes. La instrucción a usar es *AND dd* (E6h 0 230) que hace un *AND* del contenido del acumulador con el *byte* que venga detrás de la instrucción.

Por último hay que añadir que al final de la instrucción *RET* para que al acabar el programa devuelva el control al *BASIC*.

El listado completo se da en la figura 8 junto con sus correspondientes códigos decimales y hexadecimales. Esta vez el formato cambia bastante con respecto a los dados en los meses anteriores, ya que es la salida directa de un ensamblador. En primer lugar veremos que la columna de la izquierda indica el número de la línea, a continuación va la dirección en hexadecimal de la instrucción seguida por los códigos hexadecimales de esta. La siguiente columna (vacía en muchos casos) es la de etiquetas, que explicaremos ahora, y por último viene a instrucción y los datos de ésta en el formato normal, pero con ligeras variaciones. En primer lugar, las instrucciones con formato inmediato (el dato sigue a la instrucción en lugar de la dirección) no llevan el símbolo # delante , además si se utilizan números

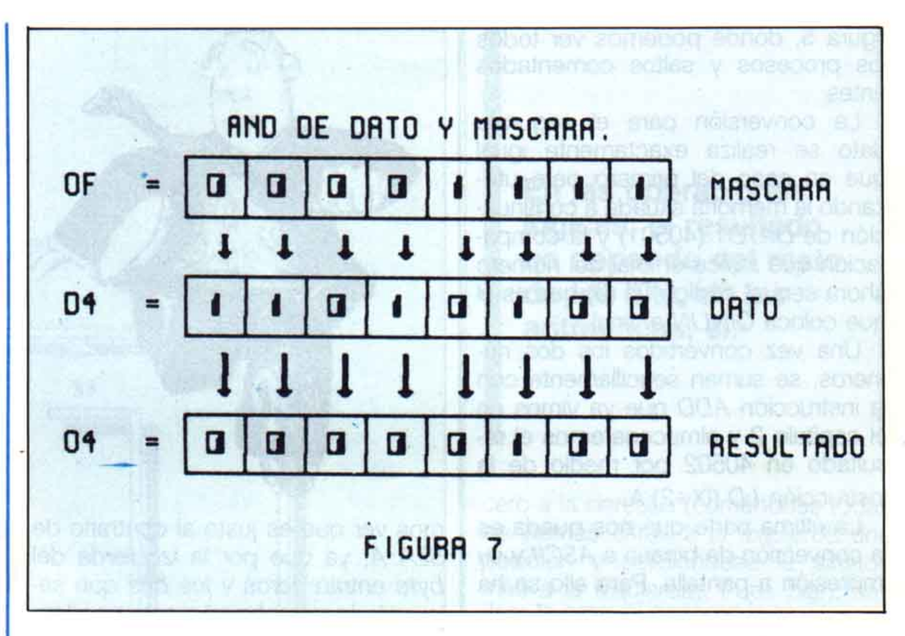


FIGURA 7

Las etiquetas se utilizan para bifurcar hacia alguna dirección de un programa sin tener que realizar cálculos engorrosos.

hexadecimales (cosa que no se ha hecho en este listado) deben ir seguidos de un «H» y si empiezan con un carácter de «A» a «F» deben ir precedidas por un cero para que no sean confundidos con etiquetas.

Por último hay que observar que en los saltos y en algunas otras instrucciones, no se ha puesto la dirección sino la susodicha etiqueta. Esto es un nombre con el que se identifica una dirección y que el programa esamblador se encarga de sustituir por ésta. Esto resulta de gran utilidad ya que nos evita el hacer cálculos engorrosos propensos a equivocaciones. Evidentemente aquellos que no usen ensamblador y hagan todo a mano no podrán usar esta facilidad.

También hay que señalar que hay determinados comandos que no generan códigos de operación, como son el *ORG, LOAD, END* o los *EQU*. Esto se debe a que son órdenes al programa ensamblador y no-instrucciones del Z-80, por lo que aquellos que ensamblen a mano la deben ignorar. Para aquellos que se hayan tecnificado diremos que el *ORG* indica la dirección donde se va a almacenar y ejecutar el programa (40000 en nuestro caso o 9C40h), que debe ir seguido siempre por un *LOAD* en la misma dirección, que hace que el ensamblador almacene los códigos hexadecimales en esa dirección. La existencia de estas dos instrucciones y no de una sóla se debe a que permiten usos complejos que, de momento, no nos interesan.

Por ahora basta saber que ambas deben ir al principio del programa y con la misma dirección. Los *EQU* asignan direcciones a las etiquetas que no se hayan definidas en la otra parte del programa. Este es el caso de la posición de memoria *DATO*1 y las de las subrutinas de la *ROM*. En cambio las direcciones usadas por los saltos internos del programa se definen en la propia dirección y no hay que definirlas aparte. Fíjese que cuando se define una etiqueta va seguida de dos puntos, pero cuando

se la llama no hace falta. Por último, el *END* debe ir después de todo lo demás y le indica al ensamblador que ahí se acaba el listado.

Protegiendo nuestros programas del BASIC

Hasta ahora hemos introducido nuestros programas en memoria sin preocuparnos de si esa zona estaba ocupada o nó, o era usada por el *BASIC*. Para evitar que se choquen entre sí y que el *BASIC* se ponga a escribir encima de nuestro programa, hay una instrucción que nos permite fijar un límite máximo de memoria a usar por el ordenador. Esta ins-



trucción es *CLEAR*. Tecleando *CLEAR* 200,39999 el ordenador nos reserva todas las direciones de memoria a partir de la 40000 (pero reservando las superiores a F37Fh para el sistema) para nuestro uso con lo que se evitan problemas. El 200 es un parámetro que indica qué espacio se reserva en *BASIC* para cadenas alfanuméricas y por tanto no nos interesa usarlo, si aquí ponemos ese valor es porque es el que adopta el MSX por defecto, pero podría ser cualquier otro que convenga. Si ejecutamos esa instrucción antes de cargar el programa, no tendremos ningún problema si funcionamos a la vez con el *BASIC*.

Con esto terminamos este mes. En el próximo número veremos como podemos hacer más pequeño y más eficiente el programa con el uso de la subrutinas, que nos permiten además disponer de diversos módulos de uso múltiples, como son los de conversión de *ASCII* a hexadecimal y viceversa que hemos visto pero expuestos de un modo más general para que puedan ser usados por otros programas.

```
 1                           ORG  40000
 2                           LOAD 40000
 3                QINLIN:    EQU  180
 4                DATO1:     EQU  40500
 5                CHPUT:     EQU  162
 6 9C40 DD21349E             LD   IX,DATO1
 7 9C44 CDB400               CALL QINLIN
 8 9C47 3E00                 LD   A,0
 9 9C49 DD7700               LD   (IX+0),A
10 9C4C DD7701               LD   (IX+1),A
11 9C4F 23                   INC  HL
12 9C50 7E                   LD   A,(HL)
13 9C51 23                   INC  HL
14 9C52 FE41      COMPA:     CP   65
15 9C54 F25B9C               JP   P,LETRA
16 9C57 D630                 SUB  48
17 9C59 1802                 JR   SUMA
18 9C5B D637      LETRA:     SUB  55
19 9C5D DD8600    SUMA:      ADD  A,(IX+0)
20 9C60 47                   LD   B,A
21 9C61 7E                   LD   A,(HL)
22 9C62 23                   INC  HL
23 9C63 FE2B                 CP   "+"
24 9C65 280D                 JR   Z,SEGUN
25 9C67 CB20                 SLA  B
26 9C69 CB20                 SLA  B
27 9C6B CB20                 SLA  B
28 9C6D CB20                 SLA  B
29 9C6F DD7000               LD   (IX+0),B
30 9C72 18DE                 JR   COMPA
31 9C74 78        SEGUN:     LD   A,B
32 9C75 DD7700               LD   (IX+0),A
33 9C78 7E                   LD   A,(HL)
34 9C79 23                   INC  HL
35 9C7A FE41      COMPA2:    CP   65
36 9C7C F2839C               JP   P,LETRA2
37 9C7F D630                 SUB  48
38 9C81 1802                 JR   SUMA2
39 9C83 D637      LETRA2:    SUB  55
```

```
40 9C85 DD8601    SUMA2:     ADD  A,(IX+1)
41 9C88 47                   LD   B,A
42 9C89 7E                   LD   A,(HL)
43 9C8A 23                   INC  HL
44 9C8B FE00                 CP   0
45 9C8D 280D                 JR   Z,TOTAL
46 9C8F CB20                 SLA  B
47 9C91 CB20                 SLA  B
48 9C93 CB20                 SLA  B
49 9C95 CB20                 SLA  B
50 9C97 DD7001               LD   (IX+1),B
51 9C9A 18DE                 JR   COMPA2
52 9C9C DD7001    TOTAL:     LD   (IX+1),B
53 9C9F DD7E00               LD   A,(IX+0)
54 9CA2 DD4601               LD   B,(IX+1)
55 9CA5 80                   ADD  A,B
56 9CA6 DD7702               LD   (IX+2),A
57 9CA9 DD7E02    IMPRE:     LD   A,(IX+2)
58 9CAC CB3F                 SRL  A
59 9CAE CB3F                 SRL  A
60 9CB0 CB3F                 SRL  A
61 9CB2 CB3F                 SRL  A
62 9CB4 FE0A                 CP   10
63 9CB6 F2BD9C               JP   P,SUMLET
64 9CB9 C630                 ADD  A,48
65 9CBB 1802                 JR   SACA
66 9CBD C637      SUMLET:    ADD  A,55
67 9CBF CDA200    SACA:      CALL CHPUT
68 9CC2 DD7E02               LD   A,(IX+2)
69 9CC5 E60F                 AND  15
70 9CC7 FE0A                 CP   10
71 9CC9 F2D09C               JP   P,SUMLE2
72 9CCC C630                 ADD  A,48
73 9CCE 1802                 JR   SACA2
74 9CD0 C637      SUMLE2:    ADD  A,55
75 9CD2 CDA200    SACA2:     CALL CHPUT
76 9CD5 C9                   RET
77                           END
```

Rincón del lector

TRADUCIR PROGRAMAS DE UN ORDENADOR AL MSX

En la mayor parte de revistas e incluso en un curso de ordenadores por fascículos, la mayor parte de los listados de programas son para el ZX Spectrum o Commodore. ¿Hay alguna forma de traducirlo para MSX? Pues algunos de ellos, cuando sepamos introducir programas, nos interesan, por ejemplo, lecciones de geografía, geometría y algún juego.

Aureo Garvin López
Barcelona.

Efectivamente, es posible "traducir" programas escritos en un BASIC a otro, pero explicarlo como respuesta a una carta sería un tanto difícil y complicado. Desde luego, la pregunta es tema para desarrollarlo en un artículo independiente. Hay que decir que, así como el BASIC se puede traducir casi directamente de unos ordenadores a otros, se debe poner cuidado al realizar conversiones entre POKEs y PEEKs, por ejemplo. A su vez, los caracteres gráficos hay que definirlos nuevamente, también habrá que redefinir las pantallas de presentación. En suma es un trabajo laborioso que, como hemos indicado anteriormente, lo trataremos en un artículo; pero, para que observe la magnitud que es alterar un programa de cualquiera de estos ordenadores, por el momento diremos que las instrucciones del Spectrum; BORDER, PAPER e INK, en el MSX equivalen a COLOR i,j,k y que la realización de gráficos en el Commodore se hace empleando numerosas sentencias POKEs, mientras que en el MSX, basta con un par de comandos, como CHR$ y SPRITE$ para hacer toda esa labor.

FALTA DE MEMORIA PARA INTRODUCIR UN PROGRAMA

En el programa «COCHES LOCOS» mi ordenador, un SONY de 16K da el error OUT OF MEMORY IN 60. Mi pregunta es, ¿se trata de un error del programa o se debe a la memoria de mi ordenador?

José Vicente Planells
Valencia

Efectivamente su ordenador no dispone de memoria suficiente para ejecutar este programa ya que requiere un mínimo de 32K RAM para funcionar correctamente. Ha sido un lapsus que procuraremos remediar en números futuros.



DIBUJAR EN UNA IMPRESORA

He adquirido recientemente una impresora MSX y no se como hacer para que me imprima un círculo o una línea (mediante las instrucciones LINE CIRCLE ect).

David Forcada
Barcelona

Lo que usted pretende es totalmente imposible en una impresora normal, aunque si dispone de una impresora tipo plotter podrá realizar estos gráficos utilizando comandos LPRINT pero nunca mediante las instrucciones graficas.

DIRECTOR:
Juan Arencibia
COORDINADOR EDITORIAL:
J. Ignacio Rey
REDACCION:
Fernando García, Santiago Gala, Ricardo García, Teresa Aranda, Francisco Mancera.
DISEÑO:
Ricardo Segura y Benito Gil.
Editada por
PUBLINFORMATICA S.A.
PRESIDENTE:
Fernando Bolín.
PUBLINFORMATICA
GERENTE DE CIRCULACION Y VENTAS:
Luis Carrero.
PRODUCCION:
Miguel Onieva.
DIRECTOR DE MARKETING:
Antonio Gonzalez.
SERVICIO AL CLIENTE:
Julia Gonzalez.
Tel. 733 79 69
ADMINISTRACION:
Miguel Atance,
JEFE DE PUBLICIDAD:
María José Martín.
DIRECCION Y REDACCION:
C/ Bravo Murillo, 377 - 5° A
Tel. 733 74 13
28020 MADRID
PUBLICIDAD Y ADMINISTRACCION:
C/ Bravo Murillo 377 - 3° E
Tel. 733 96 62-96
Publicidad en Madrid:
Fernando Hernando
Publicidad en Barcelona:
Olga Martorell
C/ Pelayo, 12
Tel. (93) 301 47 00 Ext. 27-28.
08001 BARCELONA
Deposito Legal: M. 16.755-1985
Impreso en Héroes S.A.
C/ Torrelara, 8. 28016-MADRID.
Distribuye:
S.G.E.L. Avda. Valdelaparra s/n.
Alcobendas (Madrid).

SUSCRIPCIONES:
Rogamos dirija toda la correspondencia relacionada con suscripciones a:
MSX
EDISA: Tel. 415 97 12
C/López de Hoyos, 141-5.°
28002 MADRID
(Para todos los pagos reseñar solamente MSX)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
MSX
C/Bravo Murillo, 377-5.° A
Tel. 733 74 13 28020 MADRID

Si deseas colaborar en MSX remite tus artículos o programas a Bravo Murillo 377, 5.° A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.

Programas Sony para ordenadores MSX

A la orden.


Monkey Academy


Países del Mundo-1


Países del Mundo-2


Computador Adivino


Computer Billiards


The Snowman


Cubit


Character Collection


Stop the express (Para el Tren)


Hustler (Billar Americano)


Data cartridge


Quinielas y Reducciones


Home Writer


Sparkie


Aprendiendo Inglés-1


Binary Land


Creative Greetings


Aprendiendo Inglés-2


Antartic Adventure


Mastermind


Contabilidad Personal


Athletic Land


E.I.


Ficheros


El Ahorcado


Dorodon


La Pulga


Cosmos


Control de Stocks


Battle Cross


Mouser


Crazy Train


Ali baba


Juno First


Car Jamboree


Tutor


Track and Field-1 (olimpiadas)


Blackjack


Track and Field-2 (olimpiadas)


Driller Tanks (Tanque Destructor)


Sonygraph


Ninja (El Samurai)




Les Flics

Y muchos más títulos



Ordenador Doméstico



Para lo que guste ordenar.



DYNADATA
I N F O R M A T I C A

presenta su ordenador



El ordenador DYNADATA-MSX, fabricado por DAEWOO, representa la unión tecnológica entre Oriente y Occidente.

DAEWOO, el gigante coreano, con una implantación mundial en sectores como electrónica de consumo, el naval, automóvil, textil, banca, financiero, etc., ha confiado la distribución de sus productos de MSX en exclusiva a DYNADATA.

DYNADATA, primera firma que introdujo el nuevo standard mundial MSX en España, pone de relieve el orgullo que representa haber sido seleccionada como representante de DAEWOO en el mercado español.

DAEWOO	
MICROPROCESADOR	Z80 A
MEMORIA PRINCIPAL	
RAM	64 Kbyte 16 Kbyte (VRAM video)
ROM	32 Kbyte (MSX-BASIC)
PANTALLA	24 líneas × 40 columnas en texto 256 × 192 pixels resolución gráfica 16 colores Video Compuesto y RF
SONIDO	8 octavas, 3 canales
TECLADO	73 teclas, 5 teclas de funciones (10 funciones)
SALIDAS	Casette 1200/2400 baudios Paralela Centronics para impresora Joystick Conector de expansión Slot para cartuchos
LENGUAJE	MSX-BASIC
SISTEMA OPERATIVO	MSX-DOS CP/M-80 (opcional)

DYNADATA-MSX, supone un gran paso de cara al usuario, ya que puede disponer de un ordenador de características profesionales a precio de ordenador doméstico y con la ventaja de estar encuadrado en el nuevo standard MSX.

DYNADATA-MSX, un ordenador concebido con nuevos criterios en cuanto a tecnología y ergonomía, sobre todo a nivel de comodidad y sensibilidad en el teclado de carácter profesional, que le diferencia de sus inmediatos competidores.

DYNADATA-MSX incorpora el transformador en el interior del ordenador.
Soporta una o dos unidades de diskettes de 5 1/4" ó 3 1/2".
Se puede trabajar bajo sistemas operativos CPM y MSX-DOS, lo cual abre un campo muy amplio en cuanto a disponibilidad de SOFTWARE.

DYNADATA-MSX, imbatible en el mercado en cualquiera de sus configuraciones.

DYNADATA-MSX, con Monitor de fósforo verde	69.900 ptas.
DYNADATA-MSX, con Monitor de color	105.000 ptas.
Unidad de Cassette	6.800 ptas.
Unidad Lectora de Diskette-5 1/4", doble cara, doble densidad	56.800 ptas.
Quick Disk-3 1/2"	32.700 ptas.

Otros periféricos disponible: PLOTTER, JOYSTICK, IMPRESORAS.
SOFTWARE disponible: entretenimiento, educativo, utilidad y gestión.



VEANOS EN S.I.M.O. 85
Stand E 38, Pabellon XI
Planta Superior

Sor Angela de la Cruz, 24 - 28020 Madrid. Teléfs. (91) 279 21 85 - 279 28 01 - 270 01 93 Telex 44619 DYNA
DELEGACION: Aribau, 61, entlo. 08011 Barcelona. Teléfs. (93) 254 73 04 - 254 73 03

resorte